# TIME EVOLUTION IN QUANTUM DOTS

## Using the Multiconfiguration Time-Dependent Hartree-Fock Method

by

**Sigve Bøe Skattum**

**THESIS**

for the degree of

**MASTER OF SCIENCE**

**(Master in Computational Physics)**


Faculty of Mathematics and Natural Sciences
University of Oslo

June 2013

# Acknowledgements

I would like to thank everyone who has contribute in some way to my thesis. First, I would like to thank my thesis advisor, Morten Hjorth-Jensen, you are always available, inspiring and encouraging.

Thanks to Simen Kvaal for taking the time to discuss time-dependent methods in quantum mechanics. There are few people with your level of insight into these methods - and especially thanks for our discussions about the MCTDHF-method.

To my fellow students Jørgen Høgberget, Sarah Reiman, Karl Leikanger, Arnfinn Paulsrud, Sven-Arne Dragly, and Milad Hobbi Mobarhan. Thank you for all the good discussions, all the inspiration you have given me, and for all the fun we have had!

Sigve Bøe Skattum
Oslo, June 2013

# Contents

# Chapter 1

# Introduction

The aim of this thesis is to study numerical approaches for solving the time-dependent Schrödinger equation for systems of electrons confined in quantum dots (QD) [1]. Time evolution in quantum physics is a field of study where there still is much uncovered science. Historically a lack of computing power and efficient algorithms have made it difficult to perform time-dependent calculations on non-trivial quantum systems. With the arrival of more powerful computers the possibilities for time evolusion has increased. The standard approach has been to solve the full time-dependent Schrödinger equation. However, this approach, while numerically exact within the model space, scales exponentially with the number of degrees of freedom, and the increase in computing power is still not enough. Using approximate methods the exponential scaling can be avoided - or at least staggered. To study time-dependent methods we first have to examine time-independent methods of solving the quantum many-body problem, as most time-dependent methods are based on the principles of their time-independent counterparts. To solve the time-independent many-body problem, we have methods like Hartree-Fock, Configuration Interaction [2], Coupled Cluster [3], Variational Monte Carlo [4], Diffusion Monte Carlo [5] and Density-Functional Theory [6]. All these methods have different strengths and weaknesses, e.g., the Configuration Interaction is numerically exact[1], but comes at a huge computational cost, Density Functional Theory can simulate large systems, but the functional form does not always describe the system correctly. Some of the methods above have a time-dependent version, or sometimes a combination of the methods mentioned are used to create a time-dependent method.

One method, the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) - the *de facto* method for time evolution [7], is the one we will focus on

[1] Within a chosen model space.

in this thesis. The MCTDHF method combines ideas from Time-Dependent Configuration Interaction (TDCI) [8] and Time-Dependent Hartree-Fock (TDHF) [9], into a new method.

Using the MCTDHF method we will explore the time evolution of quantum dots. The study of quantum dots is an active field which has gotten much attention lately, but in terms of time-dependent computations little has previously been done. The purpose of this thesis is to efficiently solve the MCTDHF equations of motion by developing a C++ program for time evolution. The implementation is described, and the code is shown to work by reproducing known published results. The systems we explore are mostly of academic interest, but as the code is made to be highly modular the study of more realistic systems are possible. The ground state is found by imaginary time propagation and dynamics is studied by applying time-dependent electric field to the system. Next, we study the double quantum dot in one- and two-dimensions. The double quantum dot is modeled by two potential wells in close vicinity of one another. Dynamics are studied by applying electric fields. The double quantum dot is used, for example, as qubits in quantum computers [10], a highly active field of research. However, this requires an interaction with a magnetic field - a feature not yet implemented, but the plans are there.

The aim of this thesis is to study the MCTDHF method, and apply it to systems of quantum dots. The quantum dots are highly modular systems which, by tuning the confinement strength, we can use to simulate both highly correlated systems and systems with only weak correlations. This way we can study correlation effects in quantum dots. But another aspect is to examine how the MCTDHF method manages for highly correlated systems. Methods like the Time-Dependent Hartree-Fock is known to behave badly for highly correlated systems. We will examine how the MCTDHF is affected at different degrees of correlation and compare the results with TDHF. The convergence properties of MCTDHF will also be studied.

As a part of the research done for this thesis an assorted number of quantum many-body methods were studied and numerically implemented. Methods programmed are Hartree-Fock, Configuration Interaction, Variational Monte Carlo, Diffusion Monte Carlo and Time-Dependent Configuration Interaction. Some of the methods are used as a basis for comparison of results later shown. If the reader is interested, some of these implementations are made freely available at `https://github.com/sigvebs` under a GPL license. The MCTDHF code developed as part of this thesis is also found at this address.

Work done in time evolution of quantum systems by earlier masters students are:

- 'A critical study of the finite difference and finite element methods for the time dependent Schrödinger equation' [11] by Simen Kvaal.
- 'Time dependent study of quantum dots' [12] by Jakob Kryvi.

And the PhD dissertations

- 'Quantum Control of Strongly Coupled Dynamics in Few Component Systems' [13] by Lene Sælen.
- 'PyProp - a Python Framework for Propagating the Time Dependent Schrödinger Equation' [14] by Tore Birkeland.

However, none of these studied the MCTDHF method. Therefore, much work went into understanding and developing a MCTDHF code from scratch. To write this type of program code without any reference code, and very little documentation on the implementation, is a time-consuming process. A very helpful article on the formalism of MCTDHF is found in reference [15]. A general introduction to the MCTDH-method can be found in reference [7] or the book 'Multidimensional Quantum Dynamics' [16], which also introduces MCTDHF and MCTDHB and mixtures of fermions and bosons.

**A Note on Computer Programming**

A huge part of the workload was the development of code for solving quantum many-body problems. For modularity and expansion options the C++ was chosen as the programming language. C++ is a highly efficient, low level language. However, there are certain high level operations C++ is not efficient at. For such problems the programming language Python is used. A basic knowledge of object-oriented programming is assumed, and a familiarity with C++ makes reading the implementation chapters easier, however, the implementation is meant to describe the general idea. If you are unfamiliar with C++ and/or object-orientation, see, for example, reference [17] for a good introduction.

For matrix operations and linear algebra the excellent C++ library Armadillo [18] is used. Armadillo is a template library that calls upon the highly optimized linear algebra functions in LaPack [19] and Blas [20].

## 1.1 Thesis Structure

The thesis is split into three parts: theory, implementation, and results.

### Part I: Theory

Part I introduces the theoretical framework used in this thesis.

- Chapter 2 reviews basic quantum mechanics with an emphasis on the formalism used. The single-particle notation is presented, and an example of the harmonic oscillator is shown.
- Chapter 3 introduces the theoretical framework used in quantum many-body theory. Especially the concepts of second quantization and Slater determinants are important.
- Chapter 4 presents different many-body methods for solving the time-independent Schrödinger equation.
- Chapter 5 introduces the formalism used for time evolution and presents some methods for time propagation. Possible methods are Time-Dependent Configuration Interaction (TDCI), Time-Dependent Density Functional Theory (TDDFT), Time-Dependent Hartree-Fock (TDHF), the Orbital-Adaptive Time-Dependent Coupled-Cluster (OATCC) method, the Multiconfiguration Time-Dependent Hartree (MCTDH) and at last the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) method.
- Chapter 6 presents the Multiconfiguration Time-Dependent Hartree-Fock method in detail. The MCTDHF method is thoroughly derived and explained. This method should be understood before reading Part II.
- Chapter 7 gives a short introduction to how an initial state is prepared through imaginary time propagation.

### Part II: Implementation and Verification

Part II describes the numerical implementation of the MCTDHF method. The different choices are described in detail.

- Chapter 8 describes the general development and implementation of MCTDHF.
- Chapter 9 discusses different representations of differential operators and their numerical implementation.
- Chapter 10 reviews different numerical time-integration schemes used to solve the MCTDHF equations of motion.

- Chapter 11 discusses the implementation of the mean fields, and possible optimization schemes.
- In Chapter 12 the MCTDHF implementation is tested by reproduces analytic and published results to verify the validity of the code.

**Part III: Systems and Results**

- Chapter 13 gives an introduction to quantum dots, both a qualitative description and the theoretical framework. The form of the Hamiltonian is presented together with possible potentials used to study confinement. Time dependent electromagnetic fields are discussed.
- Chapter 14 present the numerical results. The MCTDHF method is used to calculate properties of interest for single and double quantum dots in one- and two-dimensions.
- Chapter 15 summarizes and concludes this thesis.

# Part I

# Theory

# Chapter 2

# Quantum Mechanics

Quantum physics is the study of nature at the atomic scale, and forms the foundation of modern physics. In the quantum world phenomena behave in a non-classical way, making it very non-intuitive. Some examples are the wave-particle dualism, the discrete values of energy, and the uncertainty principle. In this chapter the basic theory of one-particle quantum mechanics is presented. It is assumed that the reader has a basic understanding of quantum mechanics as this chapter mostly presents the formalism used throughout the thesis. Most results are not derived thoroughly. For a detailed introduction to quantum mechanics, I recommend Sakurai's book 'Modern Quantum Mechanics' [21]. Reference [22] and [23] are good supplements. The material presented in this chapter is based on explanations covered in these texts.

### A Brief Historical Introduction to Quantum Mechanics

In the late 19th several experimental were performed that could not be explained within the framework of classical physics, and scientist started exploring alternative formulations. The start of the quantum era was when Max Planck published his theory of *black body radiation*. He proposed that electromagnetic radiation in cavities can only be absorbed and emitted at discrete quanta of energy $E = h\nu$, where $\nu$ is the frequency of the electromagnetic radiation and $h$ is a fundamental constant called *Planck's constant*. However, Planck insisted that this quantization was not a physical reality. Einstein took this a step further and explained, through the *photoelectric effect*, that light is composed of quantized particles of energy called *photons*. Using the theory of quantization, Bohr managed to explain the spectral lines of the hydrogen atom, and together with Rutherford create the first working model of the atom. The idea that waves could be described as particles was a big jump from classical mechanics, and later de Broglie proposed that par-

ticles can be described as *matter waves* and that waves can be described as particles, which has been shown experimentally to be correct.

## 2.1 The Fundamental Postulates of Quantum Mechanics

Every mathematical and physical theory is based upon some fundamental hypotheses that are postulated. We will now present the fundamental postulates of quantum mechanics. Each postulate is followed by a comment. The wording of the postulates is based on reference [23].

**Postulate 1:**

> *A quantum state of an isolated physical system is described by a vector in a complex, linear vector space, called the Hilbert space.*

Using Dirac's bra-ket notation, a quantum state is represented by a vector $|\Psi\rangle$ called a *ket*. To be part of a Hilbert space there must exist an inner product relating two vectors $|\psi_\alpha\rangle$ and $|\psi_\beta\rangle$ through $\langle\psi_\alpha|\psi_\beta\rangle \in \mathbb{C}$. The $\langle\cdot|$ vector is called a *bra* and forms a dual space with the ket vector. The inner product has the following properties:

- The complex conjugate of an inner product is the same as swapping the elements in the vectors:

  $$\langle\psi_\alpha|\psi_\beta\rangle = \langle\psi_\beta|\psi_\alpha\rangle^* \ ,$$

  where the asterisk, $*$, represents the mathematical operation of complex conjugation.

- The inner product is linear in the second argument:

  $$\langle\psi_\gamma|a\psi_\alpha + b\psi_\beta\rangle = a\langle\psi_\gamma|\psi_\alpha\rangle + b\langle\psi_\gamma|\psi_\beta\rangle \ ,$$

  and anti-linear in the first argument:

  $$\langle a\psi_\alpha + b\psi_\beta|\psi_\gamma\rangle = a^*\langle\psi_\alpha|\psi_\gamma\rangle + b^*\langle\psi_\beta|\psi_\gamma\rangle \ ,$$

  for $a, b \in \mathbb{C}$.

- The inner product of a vector with itself is always positive definite:

  $$\langle\psi_\alpha|\psi_\alpha\rangle \geq 0 \ .$$

Given a complete, discrete basis $\{i\}$, it must fulfill the completeness relation

$$\mathbb{1} = \sum_{i=1}^{N} |i\rangle \langle i| \ , \quad \langle i|j\rangle = \delta_{i,j} \ . \tag{2.1}$$

A state vector can be expanded in such a basis by

$$|\Psi\rangle = \sum_i \langle \Psi|i\rangle \langle i| = \sum_i c_i |i\rangle \ . \tag{2.2}$$

We can also expand the state in a continuous basis, e.g., the position space $\{x\}$. For a continuous space the completeness relation reads

$$\mathbb{1} = \int_{-\infty}^{\infty} |x\rangle \langle x| \ dx \ , \tag{2.3}$$

and the $|\Psi\rangle$ can be written as

$$|\Psi\rangle = \int_{-\infty}^{\infty} \Psi(x) |x\rangle \ dx, \quad \Psi(x) = \langle x|\Psi\rangle \ . \tag{2.4}$$

A state is said to be *normalized* if the inner product with itself is one:

$$\langle \Psi|\Psi\rangle = 1 \ . \tag{2.5}$$

**Postulate 2:**

> *A physical observable, A, of a system is a always associated with a Hermitian operator, $\hat{A}$. The eigenstates of $\hat{A}$ defines a complete, orthonormal set of vectors.*

A Hermitian operator has the following properties

$$\langle \psi_\alpha|\hat{A}|\psi_\beta\rangle = \langle \psi_\alpha \hat{A}|\psi_\beta\rangle = \langle \psi_\alpha|\hat{A}\psi_\beta\rangle \ , \tag{2.6}$$

when acted upon by two states $|\psi_\alpha\rangle$ and $|\psi_\beta\rangle$, implying that $\hat{A} = \hat{A}^\dagger$, where $\dagger$ means transposed and complex conjugated. An eigenvalue equation is defined as

$$\hat{A} |a_i\rangle = a_i |a_i\rangle \ , \tag{2.7}$$

where $a_i$ is an eigenvalue and $|a_i\rangle$ is an eigenstate of $\hat{A}$. The set of all eigenstates $\{|a_i\rangle\}$ of $\hat{A}$ must form a complete set of vectors

$$\mathbb{1} = \sum_i |a_i\rangle \langle a_i| \ . \tag{2.8}$$

The spectral decompositions of a Hermitian operator is

$$\hat{A} = \sum a_i |a_i\rangle \langle a_i| \ , \quad a_i = a_i^* \ . \tag{2.9}$$

**Postulate 3:**

*The time evolution of a state vector, $|\Psi(t)\rangle$, is given by the Schrödinger equation,*

$$i\hbar\frac{d}{dt}|\Psi\rangle = \hat{H}|\Psi\rangle \ ,$$

*where $\hat{H}$ is the Hamiltonian, and $\hbar$ is Planck's reduced constant.*

The Schrödinger equation is linear in the time-derivate and is uniquely defined by the initial state $|\Psi(t_0)\rangle$. The Hamiltonian is a Hermitian operator, and can be expressed as

$$\hat{H} = \hat{T} + \hat{V} = \frac{\hat{p}^2}{2m} + \hat{V} \ , \tag{2.10}$$

where $\hat{T}$ is the kinetic operator, $\hat{p}$ is the momentum operator, $\hat{V}$ is the potential operator and $m$ is the mass.

**Postulate 4:**

*A measurement of an observable quantity, A, will always result in one of the eigenvalues, $a_i$, of the operator $\hat{A}$.*

The probability of measuring an eigenvalue, $a_i$, in a state described by $|\Psi\rangle$, is found by

$$p_i = |\langle a_i|\Psi\rangle|^2 \ . \tag{2.11}$$

The expectation value of an observable is found by

$$\langle A\rangle = \langle\Psi|\hat{A}|\Psi\rangle \ , \tag{2.12}$$

and is interpreted as the mean value of an infinite set of measurements.

**Postulate 5:**

*By measuring a value $a_i$ of the observable A, the system will collapse into the state given by the eigenstate of $a_i$, i.e., $|\Psi\rangle \rightarrow |a_i\rangle$.*

This is called the collapse of the wavefunction.

## 2.2 Basic Quantum Mechanics

Until now we have written all states as abstract state-vectors. We will now show how to represent states in coordinate space, which is more appropriate for computations. If $|\mathbf{r}\rangle$ is an eigenvector in coordinate space, it must have a corresponding eigenvalue, $\mathbf{r}$, so that

$$\hat{\mathbf{r}}\,|\mathbf{r}\rangle = \mathbf{r}\,|\mathbf{r}\rangle\,. \tag{2.13}$$

The overlap of $|\mathbf{r}\rangle$ with a state $|\Psi\rangle$ is written in a functional form:

$$\langle\mathbf{r}|\Psi\rangle = \Psi(\mathbf{r})\ , \tag{2.14}$$

giving us a spatial representation of the state. By using such a projection, we can represent the Schrödinger equation in position space. The result is

$$i\hbar\frac{\partial\Psi(\mathbf{r},t)}{\partial t} = \frac{\mathbf{p}^2}{2m}\Psi(\mathbf{r},t) + V(\mathbf{r},t)\Psi(\mathbf{r},t)\ , \tag{2.15}$$

where $\mathbf{p}$ is the momentum operator in position space and $V(\mathbf{r},t)$ is the potential. The momentum operator is actually a differential operator when represented in position space:

$$\mathbf{p} = -i\hbar\nabla\ , \tag{2.16}$$

where $\nabla$ is the gradient operator. Inserting the spatial representation of the momentum operator into the Schrödinger equation gives the more familiar form of

$$i\hbar\frac{\partial\Psi(\mathbf{r},t)}{\partial t} = -\frac{\hbar^2}{2m}\nabla^2\Psi(\mathbf{r},t) + V(\mathbf{r},t)\Psi(\mathbf{r},t)\ . \tag{2.17}$$

If the function $\Psi$ is a solution to the Schrödinger equation, it is called a *wavefunction*. Solving the Schrödinger equation for the wavefunction $\Psi$ allows us to describe the probabilities governing a quantum system. If we know the wavefunction we can use it to measure *observable* quantities,for example, the position and momentum. Usually solving Schrödinger's equation yields a set of solutions $\{\psi_i\}$. Using initial and boundary conditions, a unique solution can be found within such a set of solutions, and the full wavefunction is written as a linear combination of such solutions:

$$\Psi(\mathbf{r},t) = \sum_i c_i\psi_i(\mathbf{r},t)\ . \tag{2.18}$$

A notable difference between classical mechanics and quantum mechanics is the commutation of operators. In classical mechanics the order of observables in an equation does not matter, i.e $\mathbf{xp} = \mathbf{px}$, but in quantum mechanics this is not necessarily true.

### The Time-Independent Schrödinger Equation

Let $\Psi(\mathbf{r}, t)$ be a solution to the Schrödinger equation

$$i\hbar\frac{\partial\Psi(\mathbf{r},t)}{\partial t} = \hat{H}\Psi(\mathbf{r},t) \ . \tag{2.19}$$

If the potential, $V$, is time-independent, the solution, $\Psi(\mathbf{r}, t)$, can be separated into a spatial and a time-dependent function:

$$\Psi(\mathbf{r},t) = \psi(\mathbf{r})\phi(t). \tag{2.20}$$

The spatial solution is given by the *time-independent* Schrödinger equation

$$H\psi(\mathbf{r}) = E\psi(\mathbf{r}) \ , \tag{2.21}$$

which is an eigenvalue problem, with $\psi$ being an eigenfunction of $H$ and $E$ is the corresponding eigenvalue called the *energy*. The time-dependent function can be solved explicitly, and is found to be

$$\phi(t) = e^{-\frac{i}{\hbar}Et} \ , \tag{2.22}$$

where $E$ is the energy found by solving the time-independent Schrödinger equation.

This thesis we will deal with both the time-dependent and the time-independent Schrödinger equation. Solutions to the time-independent Schrödinger equation are called *stationary* states.

### Relations between the position and momentum representations of the wavefunction

If we expand the spatial wavefunction in momentum space, we introduced an overlap, $\langle x|p\rangle$, between the position and momentum[1]:

$$\Psi(x,t) = \int \langle x|p\rangle\langle p|\Psi\rangle \ dp = \int \langle x|p\rangle\Phi(p,t) \ dp \ , \tag{2.23}$$

and likewise the momentum wavefunction expanded in the position basis is

$$\Phi(p,t) = \int \langle p|x\rangle\langle x|\Psi\rangle \ dx = \int \langle p|x\rangle\Psi(x,t) \ dx \ . \tag{2.24}$$

Now we have to find an expression for the overlap between position and momentum. The momentum operator acting on the position operator is

$$\hat{p}\,|x\rangle = -i\hbar\frac{d}{dx}\,|x\rangle \ , \tag{2.25}$$

[1]Note: the limits of the integrals are always from $-\infty$ to $\infty$ unless otherwise stated.

and the momentum operator on momentum space is just $\hat{p}\,|p\rangle = p\,|p\rangle$. From these results we can set up the equation

$$\langle x|\hat{p}|p\rangle = -i\hbar\frac{d}{dx}\langle x|p\rangle = p\langle x|p\rangle \ , \tag{2.26}$$

by letting the momentum operator act to the left and right. This gives us a first order, separable differential equation

$$\frac{\partial}{\partial x}\langle x|p\rangle = i\frac{p}{\hbar}\langle x|p\rangle \ , \tag{2.27}$$

which, by integration, is

$$\langle x|p\rangle = e^{\frac{i}{\hbar}xp} \ . \tag{2.28}$$

Inserting the overlap into Eq. (2.23) and Eq. (2.24) yelds

$$\Psi(x,t) = \int e^{\frac{i}{\hbar}xp}\Phi(p,t)\ dp \ , \tag{2.29}$$

$$\Phi(p,t) = \int e^{-\frac{i}{\hbar}xp}\Psi(x,t)\ dp \ . \tag{2.30}$$

Observing these equations one recognize that $\Phi$ is the Fourier transform of $\Psi$. These results can be generalized to three dimensions:

$$\Psi(\mathbf{x},t) = \int e^{\frac{i}{\hbar}\mathbf{x}\cdot\mathbf{p}}\Phi(\mathbf{p},t)\ d\mathbf{p} \ , \tag{2.31}$$

and

$$\Phi(\mathbf{p},t) = \int e^{-\frac{i}{\hbar}\mathbf{x}\cdot\mathbf{p}}\Psi(\mathbf{x},t)\ d\mathbf{x} \ . \tag{2.32}$$

From these results we can show that the position operator acting on a momentum state is

$$\langle p|\hat{x}|\Psi\rangle = \int \langle p|x\rangle x\langle x|\Psi\rangle\ dx = \int xe^{-\frac{i}{\hbar}xp}\Psi(x)\ dx = i\hbar\frac{\partial}{\partial p}\Phi \ . \tag{2.33}$$

To know the transition between position and momentum space is important. Using a Fourier transform makes it easy to go from the position representation to the momentum representation. An example why we would want to do such a transformation is that the momentum operator is in position space a differential operator, but in momentum space it is diagonal in a matrix representation. This is something we will exploit when computing the kinetic operator. Going between representations can also highlight physical attributes of a system.

## 2.3 The Quantum Harmonic Oscillator

The harmonic oscillator potential is one of the most used potentials in quantum mechanics. It is a simple system that shows important and fundamental operations used in quantum mechanics. The solutions to the harmonic oscillator is also used as a basis for more complex computations.

The form of the harmonic oscillator potential is found by performing a Taylor expansion to second order around a local minimum of a general potential, and is

$$V(x) \approx V(x_0) + \left.\frac{\partial V(x)}{\partial x}\right|_{x=x_0} (x - x_0) + \frac{1}{2} \left.\frac{\partial^2 V(x)}{\partial x^2}\right|_{x=x_0} (x - x_0)^2 \ . \quad (2.34)$$

Since we are modeling the potential around a minimum the first derivative must be zero. The constant can be disregarded without loss of generality. This leaves us with

$$V(x) \approx \frac{1}{2} m\omega^2 (x - x_0)^2 \ , \quad (2.35)$$

where we have used the particle mass $m$ and the arbitrary oscillator frequency $\omega$ to denote the constant in front of the second derivative. For the sake of simplicity we will set $x_0 = 0$. The resulting potential is

$$\hat{V} = \frac{1}{2} m\omega^2 \hat{x}^2 \ , \quad (2.36)$$

For one electron trapped in a harmonic well the time independent Schrödinger equation reads

$$\hat{H} \left|\psi\right\rangle = E \left|\psi\right\rangle \quad (2.37)$$

with $\hat{H}$ being

$$\hat{H} = \frac{\hat{p}^2}{2m} + \frac{1}{2} m\omega^2 \hat{x}^2 \ . \quad (2.38)$$

Having an expression for the Hamiltonian we can set up the Schrödinger equation:

$$\frac{\hat{p}^2}{2m} \left|\psi\right\rangle + \frac{1}{2} m\omega^2 \hat{x}^2 \left|\psi\right\rangle = E \left|\psi\right\rangle \ , \quad (2.39)$$

The usual way of solving such an equation is to split it into two first order differential equations:

$$\left[\left(\sqrt{\frac{m}{2}}\omega\hat{x} - i\frac{\hat{p}}{\sqrt{2m}}\right)\left(\sqrt{\frac{m}{2}}\omega\hat{x} + i\frac{\hat{p}}{\sqrt{2m}}\right) - \frac{i\omega}{2}[\hat{x}, \hat{p}]\right] \left|\psi\right\rangle = E \left|\psi\right\rangle \ , \quad (2.40)$$

where the last term comes in due to the non-commutative nature of quantum mechanical operators: for an operator $A$ and $B$, $AB$ is not necessarily equal

to $BA$, the operators are said to be *commuting*. In the case of the position and momentum operator, the commutation relation is

$$[\hat{x}, \hat{p}] = i\hbar \ . \tag{2.41}$$

Inserting the commutation relation into the Schrödinger equation, gives us after some reorganization

$$\hbar\omega \left[ \sqrt{\frac{m\omega}{2\hbar}} \left( x - i\frac{\hat{p}}{m\omega} \right) \sqrt{\frac{m\omega}{2\hbar}} \left( \hat{x} + i\frac{\hat{p}}{m\omega} \right) + \frac{1}{2} \right] |\psi\rangle = E\,|\psi\rangle \tag{2.42}$$

$$\hbar\omega \left[ a^\dagger a + \frac{1}{2} \right] |\psi\rangle = E\,|\psi\rangle \ , \tag{2.43}$$

where we have defined the *excitation operator* $a^\dagger$ and the *de-excitation operator* $a$. The excitation operator is defined as

$$a^\dagger = \sqrt{\frac{m\omega}{2\hbar}} \left( \hat{x} - i\frac{\hat{p}}{m\omega} \right) \ . \tag{2.44}$$

Its Hermitian conjugate is called the de-excitation operator, and is defined as

$$a = \sqrt{\frac{m\omega}{2\hbar}} \left( \hat{x} + i\frac{\hat{p}}{m\omega} \right) \ . \tag{2.45}$$

Using the definition of the excitation and de-excitation operators we find the commutation relation:

$$[a^\dagger, a] = 1 \ . \tag{2.46}$$

Using this commutation relation we can write Eq. (2.43) as

$$\hbar\omega \left[ aa^\dagger - \frac{1}{2} \right] |\psi\rangle = E\,|\psi\rangle \, . \tag{2.47}$$

Imagine that $|\psi_n\rangle$ is the $n$-th eigenstate of $\hat{H}$ with energy $E_n$, and that we want to check whether $a^\dagger\,|\psi_n\rangle$ is an eigenstate:

$$\hat{H}a^\dagger\,|\psi_n\rangle = \hbar\omega \left[ a^\dagger a a^\dagger + \frac{a^\dagger}{2} \right] |\psi\rangle \tag{2.48}$$

$$= a^\dagger \hbar\omega \left[ aa^\dagger + \frac{1}{2} \right] |\psi\rangle \ . \tag{2.49}$$

Inserting Eq. (2.47) gives us

$$\hat{H}a^\dagger\,|\psi_n\rangle = [E_n + \hbar\omega]\, a^\dagger\,|\psi\rangle \ , \tag{2.50}$$

and we see that $a^\dagger |\psi\rangle$ is indeed an eigenstate of $\hat{H}$ with energy $E_n + \hbar\omega$. Likewise we find that $a |\psi_n\rangle$ yiels

$$\hat{H} a |\psi_n\rangle = [E_n - \hbar\omega] a |\psi_n\rangle \ . \tag{2.51}$$

So both $a^\dagger$ and $a$ acting on an eigenstate of $\hat{H}$ results in a new eigenstate, with $a^\dagger$ heightening the energy, $a$ lowering it, and the change in energy is always $\hbar\omega$. For there to be any stable solution to the problem there must be a lower limit on the energy, such that

$$a |\psi_0\rangle = 0 \ . \tag{2.52}$$

From this relation we can find an expression for the ground state:

$$a |\psi_0\rangle = \sqrt{\frac{m\omega}{2\hbar}} \left( \hat{x} + i \frac{\hat{p}}{m\omega} \right) |\psi_0\rangle = 0 \ . \tag{2.53}$$

Taking the projection into the coordinate representation yields

$$\left( \hat{x} + \frac{\hbar}{m\omega} \frac{d}{dx} \right) \psi_0(x) = 0 \ . \tag{2.54}$$

Solving this differential equation results in the following expression for the ground state:

$$\psi_0(x) = \left( \frac{m\omega}{\pi\hbar} \right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} \ . \tag{2.55}$$

If we wanted the momentum representation of the ground state, we could now just perform a Fourier transformation of the above expression. In general, the solution to the Harmonic oscillator problem is on the form

$$\psi_n(x) = \frac{1}{\sqrt{2^n \, n!}} \left( \frac{m\omega}{\pi\hbar} \right)^{1/4} e^{-\frac{m\omega x^2}{2\hbar}} H_n \left( \sqrt{\frac{m\omega}{\hbar}} x \right) \ , \tag{2.56}$$

where $H_n(x)$ is the $n$-th Hermite polynomial and $n = 0, 1, 2, \cdots$. The energy numbers, $\{n\}$, are called *quantum numbers*.

# Chapter 3

# Quantum Many-Body Theory

In this chapter the one-body formalism introduced in Chapter 2 is extended to systems of many-particles. We will describe systems of identical particles; either *bosons* or *fermions*. The relations are shown for fermions, but in most cases they will, more or less, apply to bosons. We will also review the formalism for describing many-body quantum mechanics. Most notable is the *second quantization* formalism and the use of *Slater determinants* to express a wavefunction. For more details on many-body quantum mechanics see, for example, 'Many-particle Theory' [24] by Gross *et al* and 'Many-Body Methods in Chemistry and Physics' [25] by Shavitt and Bartlett. For some of the subject, like the reduced density formalism, reference [23] is recommended.

## 3.1 The Many-Body Problem

Let us consider the isolated $N$-particle system described by the wavefunction $\Psi(\mathbf{r}_1, \cdots, \mathbf{r}_N)$. The many-body Hamiltonian is defined as

$$\hat{H} = \hat{T} + \hat{V} \ , \tag{3.1}$$

where $\hat{T}$ is the kinetic operator and $\hat{V}$ is the potential operator. Both operators are now acting on all the particles. In the case of the kinetic operator, we can write is as

$$\hat{T} = \sum_{i=1}^{N} \hat{t}_i, \tag{3.2}$$

where $\hat{t}_i$ is the one-body kinetic operator acting only on particle $i$, defined as the one-body kinetic operator seen in Chapter 2. The potential term is slightly more complex as a potential can act on more than one particle at a

time. For our systems we will maximally use a two-body form - typically the Coulomb interaction between electrons. We write the potential operator as

$$\hat{V} = \sum_{i=1}^{N} v_i + \hat{V}_I, \tag{3.3}$$

where the first term is the one-body potential, and $\hat{V}_I$ is the two-body term, which is called the *interaction* term. We can now rewrite the Hamiltonian in terms of one and two-body operators:

$$\hat{H} = \hat{H}_0 + \hat{H}_I \ , \tag{3.4}$$

where

$$\hat{H}_0 = \sum_{i=1}^{N} \hat{h}_i = \sum_{i=1}^{N} \left(\hat{t}_i + \hat{v}_i\right) \ , \tag{3.5}$$

is the sum of all one-body Hamiltonians. The interaction Hamiltonian can be written as

$$\hat{H}_I = \hat{V}_I = \frac{1}{2} \sum_{i,j}^{N} \hat{v}_{ij} \ . \tag{3.6}$$

Here $\hat{v}_{ij}$ is the two-body operator describing the interaction between particle $i$ and $j$.

The simplest approximation to a many-body wavefunction is to use the product of one-body wavefunctions to create the many-body wavefunction:

$$\Phi_H(\mathbf{r}_1, \cdots, \mathbf{r}_N) = \phi_1(\mathbf{r}_1) \cdots \phi_N(\mathbf{r}_N) \tag{3.7}$$

where $\phi_i(\mathbf{r}_i)$ is the one-body wavefunction describing the $i$-th particle. This approximation of the wavefunction is called the *Hartree*-wavefunction. Note that the Hartree product does not fulfill the symmetries needed to describe fermions.

## 3.2 Systems of Identical Particles

Imagine two electrons separated by a large distance, moving towards one another[1]. Both electrons are identical, and described by a one-body wavefunction. At one point their wavefunctions will start overlapping. When this happens, there is no longer any way identifying which electron is which, and we can no longer describe one without the other - they have become

[1] We are ignoring spin for now.

correlated. Electrons are examples of identical particles, or sometimes called indistinguishable particles.

For systems containing identical particles the expectation value of an observable, $\hat{B}$, must be the same if two identical particles exchange positions. For an $N$-particle system we have

$$\langle B\rangle = \int \Psi^\dagger(\mathbf{r}_1,\cdots,\mathbf{r}_i,\cdots,\mathbf{r}_j,\cdots,\mathbf{r}_N)\hat{B}\Psi(\mathbf{r}_1,\cdots,\mathbf{r}_i,\cdots,\mathbf{r}_j,\cdots,\mathbf{r}_N)\; d^N\mathbf{r} \tag{3.8}$$

$$= \int \Psi^\dagger(\mathbf{r}_1,\cdots,\mathbf{r}_j,\cdots,\mathbf{r}_i,\cdots,\mathbf{r}_N)\hat{B}\Psi(\mathbf{r}_1,\cdots,\mathbf{r}_j,\cdots,\mathbf{r}_i,\cdots,\mathbf{r}_N)\; d^N\mathbf{r}\ , \tag{3.9}$$

where

$$\int d^N\mathbf{r} = \int d\mathbf{r}_1 \cdots \int d\mathbf{r}_N\ . \tag{3.10}$$

To better describe the change of particles we define the *permutation* operator, $\hat{P}$, as

$$\hat{P} = \prod_k \hat{P}_{ij}, \tag{3.11}$$

where $k$ is the number of permutations. For every permutation, $k$, the permutation operator exchanges two particles:

$$\begin{aligned}&\hat{P}_{ij}\phi_{\alpha_1}(\mathbf{r}_1)\cdots\phi_{\alpha_i}(\mathbf{r}_i)\cdots\phi_{\alpha_j}(\mathbf{r}_j)\cdots\phi_{\alpha_N}(\mathbf{r}_N)\\ &= \phi_{\alpha_1}(\mathbf{r}_1)\cdots\phi_{\alpha_i}(\mathbf{r}_j)\cdots\phi_{\alpha_j}(\mathbf{r}_i)\cdots\phi_{\alpha_N}(\mathbf{r}_N)\ ,\end{aligned} \tag{3.12}$$

where we have used the Hartree-wavefunction as an example. Properties of the permutation operator:

- $\hat{P}_{ij}^2 = \mathbb{1}$..
- The operator is unitary: $\hat{P}_{ij}^{-1} = \hat{P}_{ij} = \hat{P}_{ij}^\dagger = \hat{P}_{ji}$.
- It commutes with the Hamiltonian: $[\hat{P}_{ij}, \hat{H}] = 0$.
- It has eigenvalues $\pm 1$ when acting on a state consisting of identical particles.

Depending on whether the eigenvalues are one or minus one, we say that

- $\Psi$ is *symmetric* if $P_{ij}\Psi = \Psi$, for all $i$ and $j$.
- $\Psi$ is *antisymmetric* if $P_{ij}\Psi = -\Psi$, for all $i$ and $j$.

We now introduce the antisymmetrization operator

$$\hat{A} = \frac{1}{N!} \sum_{p=0}^{N} (-1)^p \hat{P}_p \ . \tag{3.13}$$

The antisymmetrization operator creates an antisymmetric wavefunction from, e.g., a Hartree-wavefunction. The factor $N!$ is included for normalization reasons. An antisymmetric wavefunction can be written

$$\Psi(\mathbf{r}_1, \cdots, \mathbf{r}_N) = \hat{A}\Phi_H(\mathbf{r}_1, \cdots, \mathbf{r}_N) \ , \tag{3.14}$$

where $\Phi_H$ is a Hartree wavefunction.

**Bosons and Fermions**

A wavefunction describing a set of identical particles must be formed from either symmetric or anti-symmetric functions. A combination is not allowed. Systems where the total wavefunction is anti-symmetric are said to be *fermionic*, and systems described by a symmetric-wavefunction are called *bosonic*. For single-particles these symmetries are decided by their intrinsic spin. We say that particles with integer spins are bosons and particles with half integer spins are fermions. Fermions have to adhere to the *Pauli principle*, and a consequence of the Pauli principle is that two identical fermions can never occupy the same quantum state. Electrons are an example of fermions, and have spin 1/2.

It turns out that this seemingly small detail is extremely important. Bosonic and fermionic particles behave drastically differently. In this thesis we will explore systems of fermions, but most of the derivations can be generalized for bosons with minor changes.

### 3.2.1 The Fermionic Wavefunction

A natural starting point for modeling the many-body wavefunction are the single-particle wavefunctions. As a first approximation to a many-body wavefunction we can use the product of $N$ single-particle wavefunctions, where $N$ is the number of particles. We have already seen such a function, the *Hartree* wavefunction:

$$\Phi_H(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \phi_{\alpha_1}(\mathbf{r}_1)\phi_{\alpha_2}(\mathbf{r}_2)\cdots\phi_{\alpha_N}(\mathbf{r}_N) \ , \tag{3.15}$$

where $\phi_{\alpha_i}$ is a single-particle wavefunction and $\alpha_i$ is a set of quantum numbers describing that state. Remember that for fermions each $\alpha_i$ must be unique.

Using the permutation operator, we create the antisymmetric wavefunction by

$$\Phi(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \frac{1}{\sqrt{N!}} \sum_{p=0}^{N} (-1)^p \hat{P}_p \Phi_H(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) \ , \tag{3.16}$$

or more compactly using the antisymmetrization operator, $\hat{A}$:

$$\Phi_0 = \sqrt{N!}\hat{A}\Phi_H \ . \tag{3.17}$$

An equivalent way of writing Eq. (3.16) is by using a *Slater determinant*. A Slater determinant wavefunction is written as

$$\Phi_\alpha(\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_N) = \frac{1}{\sqrt{N!}} \begin{vmatrix} \phi_{\alpha_1}(\mathbf{r}_1) & \cdots & \phi_{\alpha_1}(\mathbf{r}_N) \\ \vdots & & \vdots \\ \phi_{\alpha_N}(\mathbf{r}_1) & \cdots & \phi_{\alpha_N}(\mathbf{r}_N) \end{vmatrix} . \tag{3.18}$$

The state representing the Slater determinant is uniquely described by the single-particle wavefunctions. It is common to write such a state as

$$|\Phi_\alpha\rangle = |\phi_{\alpha_1} \cdots \phi_{\alpha_N}\rangle = |\phi_{\alpha_1}\rangle \otimes \cdots \otimes |\phi_{\alpha_N}\rangle \ , \tag{3.19}$$

for an $N$-particle system. Every single-particle has its own Hilbert-space, and combined they form the *Fock*-space used to express a many-body wavefunction. Although $|\Phi_\alpha\rangle$ is not strictly a Slater determinant, we will refer to it as one throughout this thesis.

Using the Slater determinants, we have found a convenient way of expressing the many-body wavefunction for fermions. But it is not given that one Slater determinant can correctly describe a general many-body wavefunction[2]. However, the space formed by all $N$-particle Slater determinants is complete, for a set of single particle wavefunctions $\{\phi_i\}$. If we express the full wavefunction as a linear combination of all $N$-particle Slater determinants we can correctly describe the many-body wavefunction:

$$\Psi(x_1, \cdots, x_N) = \sum_\alpha C_\alpha \Phi_\alpha(x_1, \cdots, x_N), \tag{3.20}$$

where $\alpha$ is a unique set of quantum numbers and $C_i$ is an expansion coefficient.

[2] There are some approximate methods that tries, though.

## 3.3 Second Quantization

A Slater determinant is written in Fock space as

$$|\Phi_\alpha\rangle = |\phi_{\alpha_1} \cdots \phi_{\alpha_N}\rangle \ . \tag{3.21}$$

Such a Slater determinant can also be written

$$|\Phi_\alpha\rangle = a^\dagger_{\alpha_1} \cdots a^\dagger_{\alpha_N} |0\rangle \ , \tag{3.22}$$

where $|0\rangle$ is called the *vacuum* state and $a^\dagger$ is a *creation operator*. The index $\alpha_i$ refers to a set of quantum numbers used to describe a single-particle state, and $\alpha$ is the collection of all quantum numbers used in the Slater determinant, $\alpha = \alpha_1, \cdots, \alpha_N$, Such operators are very similar to the excitation operators we encountered when solving the Schrödinger equation for the harmonic oscillator potential for one particle, but instead of increasing the energy, it now creates a state. When a creation operator $a^\dagger_i$ acts on a Slater determinant, it fills state $i$ in the Slater determinants Fock space:

$$a^\dagger_i |0\rangle = |\phi_i\rangle \ . \tag{3.23}$$

If the state is already occupied, the operation returns zero:

$$a^\dagger_i |\phi_i\rangle = 0 \ . \tag{3.24}$$

To remove states we introduce the *annihilation operator*, $a$. It works the following way:

$$a_i |\phi_j\rangle = \begin{cases} |0\rangle \\ 0, \end{cases} \quad \text{for } i \neq j \ , \tag{3.25}$$

$$a_i |0\rangle = 0 \ . \tag{3.26}$$

The creation and annihilation operators are related by being their Hermitian conjugates:

$$a^\dagger_i = (a_i)^\dagger \ . \tag{3.27}$$

The following anti-commutation rules apply to the creation and annihilation operators:

$$\{a^\dagger_i, a_j\} = \delta_{ij} \ , \tag{3.28}$$

$$\{a^\dagger_i, a^\dagger_j\} = 0 \ , \tag{3.29}$$

$$\{a_i, a_j\} = 0 \ . \tag{3.30}$$

The use of creation and annihilation operators to express states is called *second quantization*. For a state containing more than one particle we must be careful with the ordering of creation and annihilation operators, due to the commutations between the operators. Using the commutation rules governing creation and annihilation operators we find that

$$a_i \left|\Phi_{a,b,\cdots,i,\cdots}\right\rangle = (-1)^{n_p} \left|\Phi_{a,b,\cdots}\right\rangle \tag{3.31}$$

$$a_i^\dagger \left|\Phi_{a,b,\cdots,}\right\rangle = (-1)^{n_p} \left|\Phi_{a,b,\cdots,i,\cdots}\right\rangle \ , \tag{3.32}$$

where $n_p$ is the number of filled states before $i$. This can be shown by writing $|\Phi_{a,b,\cdots,i,\cdots}\rangle$ in terms of creation operators and employing the commutation rules given above.

### 3.3.1 The Hamiltonian Operator in Second Quantization

We start by splitting the Hamiltonian in a one- and two-body part:

$$\hat{H} = \hat{H}_0 + \hat{H}_I \ , \tag{3.33}$$

where $\hat{H}_0$ is the one-body Hamiltonian and $\hat{H}_I$ is the two-body Hamiltonian. In second quantization the one-body operator, $\hat{H}_0$ is written as

$$\hat{H}_0 = \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle a_p^\dagger a_q \ . \tag{3.34}$$

The matrix elements are typically evaluated as integrals in position space:

$$\langle \phi_p | \hat{h} | \phi_q \rangle = \int \phi_p^\dagger(\mathbf{r}) h(\mathbf{r}) \phi_q(\mathbf{r}) \ d\mathbf{r} \ . \tag{3.35}$$

For a two-body operator, $\hat{V}_I$, the second quantization form is

$$\hat{V}_I = \frac{1}{2} \sum_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle a_p^\dagger a_q^\dagger a_s a_r \ , \tag{3.36}$$

with the matrix elements being

$$\langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle = \int\int \phi_p^\dagger(\mathbf{r}_1) \phi_q^\dagger(\mathbf{r}_2) V(\mathbf{r}_1, \mathbf{r}_2) \phi_r(\mathbf{r}_1) \phi_s(\mathbf{r}_2) \, d\mathbf{r}_1 d\mathbf{r}_2 \ , \tag{3.37}$$

if evaluated in position space. The full Hamiltonian is written as

$$\hat{H} = \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle a_p^\dagger a_q^\dagger a_s a_r \ . \tag{3.38}$$

For convenience, we define the anti-symmetric two-body matrix element as

$$\langle \phi_p \phi_q || \phi_r \phi_s \rangle = \langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle - \langle \phi_p \phi_q | \hat{V} | \phi_s \phi_r \rangle \ , \tag{3.39}$$

The Hamiltonian written using anti-symmetric two-body matrix element is

$$\hat{H} = \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle a_p^\dagger a_q + \frac{1}{4} \sum_{pqrs} \langle \phi_p \phi_q || \phi_r \phi_s \rangle a_p^\dagger a_q^\dagger a_s a_r \ . \tag{3.40}$$

## 3.4 The Density Operator

The density matrix is used to describe systems of *mixed states*. Some systems need an ensemble of states to be described correctly, instead of just one. Using an ensemble, the expectation value of an operator is the average over the entire ensemble,

$$\langle C \rangle = \sum_i p_i \langle C \rangle_{\psi_i} = \sum_i p_i \langle \psi_i | \hat{C} | \psi_i \rangle \ . \tag{3.41}$$

If a system is described by only one state, it is called a *pure state*. If it cannot be described by one state, the wavefunction is called a mixed state. If we are working with an ensembles of states, a practical operator is the density operator,

$$\rho = \sum_i p_i \left| \psi_i \right\rangle \left\langle \psi_i \right| \ . \tag{3.42}$$

The density operator describes a system where the wavefunction is in a mixed state. We can use the density operator to find the expectation value of an observable. First we write

$$\langle C \rangle = \sum_i p_i \langle \psi_i | \hat{C} | \psi_i \rangle \ . \tag{3.43}$$

Using the completeness relation of a basis $\{\phi_i\}$ yields

$$\begin{aligned} \langle C \rangle &= \sum_i p_i \sum_p \langle \psi_i | \hat{C} | \phi_p \rangle \langle \phi_q | \psi_i \rangle & (3.44) \\ &= \sum_p \langle \phi_p | \left( \sum_i p_i \left| \psi_i \right\rangle \left\langle \psi_i \right| \hat{C} \right) | \phi_p \rangle \\ &= \mathrm{Tr}(\rho \hat{A}) \ . & (3.45) \end{aligned}$$

Using this equation for the expectation value it becomes clear that the weight $p_i$ represents the probability of a state in the wavefunction:

$$\langle\Psi|\Psi\rangle = \mathrm{Tr}(\rho) = \sum_i p_i = 1 \tag{3.46}$$

$$\Rightarrow p_i = |A_i^\dagger A_i| \ , \tag{3.47}$$

for a wavefunction

$$|\Psi\rangle = \sum_i A_i \, |\psi_i\rangle \ . \tag{3.48}$$

From these relations the following properties of the density operator can be found:

$$\rho = \rho^\dagger, \quad p_i > 0, \quad \sum_i p_i = 1 \ . \tag{3.49}$$

Taking the trace over all degrees of freedom yields the number of particles:

$$\mathrm{Tr}\rho = N \ . \tag{3.50}$$

### The Density Operator for Composite Systems

The density operator is used in the description of composite systems. For a composite system, consisting of subsystem $A$ and subsystem $B$, we write the density operator as $\rho_{AB}$. The subsystems are then described by the reduced density operator, given by

$$\rho_A = \mathrm{Tr}_B(\rho_{AB}) \ , \tag{3.51}$$

for system $A$. Such a composite system can be an $N$-particle system. One quantity of special interest is the reduced *one-body density*. For systems of identical fermions it tells us the density of particles, typically the electron density. This is done by taking the partial trace over all particles except one, and solving the integral in position space,

$$\rho(\mathbf{r}_1, \mathbf{r}_1') = N \int \Psi^\dagger(\mathbf{r}_1', \mathbf{r}_2, \cdots, \mathbf{r}_N)\Psi(\mathbf{r}_1, \cdots, \mathbf{r}_N) \, d\mathbf{r}_2 \cdots d\mathbf{r}_N \ , \tag{3.52}$$

or we can rewrite it as a one-body operator in second quantization

$$\rho(\mathbf{r}_1, \mathbf{r}_1') = \sum_{pq} \langle\Psi|a_p^\dagger a_q|\Psi\rangle \phi_p^\dagger(\mathbf{r}_1')\phi_q(\mathbf{r}_1) = \sum_{pq} \rho_{pq}\phi_p^\dagger(\mathbf{r}_1')\phi_q(\mathbf{r}_1) \ . \tag{3.53}$$

Here we have defined the reduced one-body density matrix as

$$\rho_{pq} = \langle \Psi | a_p^\dagger a_q | \Psi \rangle \ . \tag{3.54}$$

Diagonalizing the reduced density matrix gives us its eigenvalues and eigenvectors. The eigenvalues are interpreted as *natural populations* and the eigenvectors are the *natural orbitals* [26]. The natural population tells us abut the importance of the natural orbitals. If an orbital has a high natural population it is important in the description of the wavefunction. Likewise, a low natural populations tells us that the orbital plays a negligible role in the description of the wavefunction.

### Degree of Correlation

A wavefunction can be written in a *canonical* form,

$$\Psi = \sum_i p_i \Phi_i, \tag{3.55}$$

where $\Phi_i$ is a Slater determinant. In contrast to an ordinary expansion of the wavefunction, the canonical form is unique, and can be used as a measure of the degree of correlation in a system. The square of the weights, $p_i^2$, represents the occupation probability, where $\sum_i |p_i|^2 = 1$. We define the average occupation probability as $\sum_i |p_i|^4$. A degree of correlation [27], $K$, is proposed to be

$$K = \sum_i \frac{1}{|p_i|^4} \ . \tag{3.56}$$

It turns out that the natural populations is this unique representation of the expansion coefficients, $|p_i|^2$. The least correlated state is a single Slater determinant, and it has a value $K = 1$. For more details, see reference [27].

## 3.5 Dimensionless Form of the Hamiltonian

For simpler equations and more efficient computations the Hamiltonian is scaled to a dimensionless form. We represent dimensionless variables with a line over them, i.e. the dimensionless Hamiltonian is denoted $\overline{H}$. Let us show an example where we scale the $N$-identical particle Hamiltonian using a factor $V_0$, based on the confining potential $\hat{V}_{conf} = V_0 \overline{V}_{conf}$. The model Hamiltonian is

$$H = T + V_I + V_{conf} \ , \tag{3.57}$$

where $T$ is the kinetic operator and $V_I$ is the interaction potential. Dividing the Hamiltonian by $V_0$ gives the dimensionless form. The coordinate vector is written as $\mathbf{r} = L_0\overline{\mathbf{r}}$, where $L_0$ is the natural length scale - a scale we can choose ourselves. The dimensionless Hamiltonian is

$$\overline{H} = \overline{T} + \overline{V}_I + \overline{V}_{conf} \ . \tag{3.58}$$

The dimensionless form of the kinetic operator is

$$\overline{T} = -\sum_i \frac{\hbar^2\nabla_i^2}{2m^*}\frac{1}{V_0} = -\sum_i \frac{\overline{\nabla}_i^2}{2}\left(\frac{\hbar^2}{L_0^2 m^* V_0}\right) \ , \tag{3.59}$$

where $m^*$ is the effective mass. For a good scaling of the kinetic operator we choose the natural length scale as

$$L_0 = \frac{\hbar}{\sqrt{m^* V_0}} \ , \tag{3.60}$$

making the kinetic operator

$$\overline{T} = -\frac{1}{2}\sum_i \overline{\nabla}_i^2 \tag{3.61}$$

Later in this thesis we will use the Coulomb interaction to model the iteration between electrons. We will now perform the dimensionless scaling as an example:

$$\overline{V}_C = \frac{V_C}{V_0} = \sum_{i<j}\frac{e^2}{4\pi\epsilon_0\epsilon_r}\frac{1}{r_{ij}}\frac{1}{V_0} = \sum_{i<j}\frac{1}{\overline{r}_{ij}}\frac{e^2}{V_0 L_0 4\pi\epsilon_r} = \sum_{i<j}\frac{\overline{\lambda}}{\overline{r}_{ij}} \ , \tag{3.62}$$

where

$$\overline{\lambda} = \frac{e^2}{V_0 L_0 4\pi\epsilon_0\epsilon_r} \ . \tag{3.63}$$

Dimensionless variables are not written using a line over them in the subsequent chapters.

### Atomic Units

In atomic physics the natural length scale, $L_0$, is set equal to the Bohr radius. The Bohr radius can be written as

$$a_0 = \frac{4\pi\epsilon_0\hbar^2}{m_e e^2} = 0.529 \text{ Å} \ . \tag{3.64}$$

The unit of energy is the Hartree, which is defined as

$$E_{Ha} = V_0 = \frac{\hbar^2}{a_0^2 m_0} = m_e \left( \frac{4\pi\epsilon_0}{\hbar e} \right)^2 = 27.211 \text{ eV} \ . \tag{3.65}$$

The dimensionless form of the kinetic operator is then

$$\overline{T} = -\frac{1}{2} \sum_i \overline{\nabla}_i^2 \tag{3.66}$$

and the dimensionless Coulomb interaction is

$$\overline{V}_I = \sum_{i<j} \frac{1}{\overline{r}_{ij}} \ . \tag{3.67}$$

The implication of such a scaling is that the following quantities are set to one:

$$m_e = e = \hbar = 4\pi\epsilon_0 = 1 \ , \tag{3.68}$$

but it is just a consequence of using a smart scaling. However, when scaling back to 'real' quantities the correct units must be inserted at appropriate places. Using the atomic unit system ensures a better scalability and simpler numerical equations. Atomic units is used unless otherwise specified.

# Chapter 4

# A Selection of Many-Body Methods

In this chapter, a review is given of methods used to solve the time-independent Schrödinger equation for the many-body problem. Methods like Configuration Interaction theory and Hartree-Fock theory are derived in greater detail as they are important for understanding the time-dependent methods introduced in Chapter 5. All the methods introduced in this chapter are approximate solutions to the time-independent Schrödinger equation for the ground state:

$$\hat{H}\,|\Psi_0\rangle = E_0\,|\Psi_0\rangle \ . \tag{4.1}$$

Here $|\Psi_0\rangle$ is the ground state and $E_0$ is the ground state energy.

## 4.1 Variational Methods

The variational principle states that

$$E_0 \leq \langle\phi|\hat{H}|\phi\rangle \ , \tag{4.2}$$

where $E_0$ is the ground state of a system described the Hamiltonian $\hat{H}$. It is easy to prove this principle by expanding $|\psi\rangle$ in a set of eigenfunctions of $\hat{H}$. The principle tells us that we can use any state, $|\phi\rangle$, that adheres to the restrictions imposed by being a wavefunction, to calculate an approximation to the ground state. This allows us to calculate an upper bound on the ground state energy of a system, if we can find a good approximation to the ground state wave function. The variational principle is the starting point for many methods. A method is called *variational* if it never to produces energies lower than the actual ground state.

## 4.2 Configuration Interaction

The Configuration Interaction (CI) [2] method approximates the time-independent Schrödinger equation for the ground state

$$\hat{H}\,|\Psi_0\rangle = E_0\,|\Psi_0\rangle \tag{4.3}$$

by expanding the ground state in a complete set of orthogonal Slater determinants $\{|\Phi_i\rangle\}$, such that

$$|\Psi_0\rangle = \sum_i C_i\,|\Phi_i\rangle \ . \tag{4.4}$$

We can now rewrite the Schrödinger equation on a matrix form

$$\mathbf{HC} = E\mathbf{C}, \tag{4.5}$$

where the matrix elements of the Hamiltonian are

$$H_{ij} = \langle\Phi_i|\hat{H}|\Phi_j\rangle \ , \tag{4.6}$$

and the $\mathbf{C}$-vector contains the expansion coefficients of the wavefunction given in Eq. (4.4). After the Hamiltonian matrix is constructed we can find its eigenvectors and eigenvalues by performing a diagonalization. The lowest eigenvalue is the best approximation to the ground state. For large, Hermitian matrices the Lanczos algorithm [28] is usually the best choice of diagonalization algorithm. The results are exact if the Slater determinant basis is complete. However, in most cases it is impossible construct a complete basis, and the configuration space must be truncated. There are several schemes used to decide the truncation of the basis. The *nuclear shell model* [29] is an example of this. The idea is to use a set of known one-particle orbitals that fills all states up to a chosen energy shell. Then all possible $N$-particle Slater determinant are created from the subspace of one-particle orbitals. The method is exact within this subspace.

The CI-method is simple and numerically easy to implement for small systems, and can also be used to find exited states. But as the numerical cost increases exponentially with the size of the problem, the method is seldom used for larger systems.

## 4.3 Hartree-Fock

In the Hartree-Fock (HF) [9] method, the $N$-particle wavefunction is approximated by a single Slater determinant, $|\Phi_{HF}\rangle$. Using one Slater determinant,

the energy is

$$\langle \Phi_{HF}|\hat{H}|\Phi_{HF}\rangle = E_{ref} = \sum_a \langle \phi_a|\hat{h}|\phi_a\rangle + \frac{1}{2}\sum_{ab}\langle \phi_a\phi_b||\phi_a\phi_b\rangle \ . \tag{4.7}$$

where $\{\phi_a\}$ are the singe-particle orbitals and $||$ means an anti-symmetric matrix element. The initial choice of orbitals are the $N$ orbitals with the lowest energy. However, these orbitals might not be the best choice. To find an optimal set of orbitals, we perform a unitary transformation of the single-particle basis:

$$|\phi_a\rangle = \sum_\alpha C_{a\alpha}\,|\phi_\alpha\rangle\,, \quad \langle \phi_\alpha|\phi_\beta\rangle = \delta_{\alpha\beta}\ . \tag{4.8}$$

Using this unitary transformation, the reference energy is

$$E_{ref} = \sum_a\sum_{\alpha\beta} C^*_{a\alpha}C_{a\beta}\langle \phi_\alpha|h_0|\phi_\beta\rangle + \frac{1}{2}\sum_{ab}\sum_{\alpha\beta\gamma\delta} C^*_{a\alpha}C^*_{b\beta}C_{a\gamma}C_{b\delta}\langle \phi_\alpha\phi_\beta||\phi_\gamma\phi_\delta\rangle \ . \tag{4.9}$$

Using variational calculus we minimize the reference energy with respect to the transformed orbitals. All the constraints on the system must be introduced in the form of Lagrangian multipliers. Without changing the energy we introduce the orthogonality constraint of the orbitals, to the energy:

$$E_{ref} = E_{ref} - \sum_a \epsilon_a \left(\sum_\alpha C^*_{a\alpha}C_{a\alpha} - \delta_{a\alpha}\right)\ , \tag{4.10}$$

where $\varepsilon_a$ is a Lagrangian multiplier. Performing a variation yields

$$\delta E - \delta\left(\sum_a \epsilon_a \sum_\alpha C^*_{a\alpha}C_{a\alpha}\right) = 0\ . \tag{4.11}$$

The degrees of freedom we can vary are the expansion coefficients, $C_{a\alpha}$, of the orbitals. Performing the variation over every coefficient:

$$\delta E = \sum_{k\alpha}\frac{\partial E}{\partial C^*_{k\alpha}}\delta C^*_{k\alpha} + \sum_\alpha \frac{\partial E}{\partial C_{k\alpha}}\delta C_{k\alpha} \tag{4.12}$$

and

$$\delta\sum_a \epsilon_a\sum_\alpha C^*_{a\alpha}C_{a\alpha} = \sum_{k\alpha}\epsilon_k\left(\delta C^*_{k\alpha}C_{k\alpha} + C^*_{k\alpha}\delta C_{k\alpha}\right)\ . \tag{4.13}$$

Separating the variations:

$$\cdots + \left(\frac{\partial E}{\partial C_{k\alpha^*}} - \epsilon_k C_{k\alpha}\right)\delta C^*_{k\alpha} + \cdots = 0 \tag{4.14}$$

$$\Rightarrow \frac{\partial E}{\partial C_{k\alpha^*}} - \epsilon_k C_{k\alpha} = 0 \ . \tag{4.15}$$

Performing the derivative gives us

$$\sum_{\beta} C_{k\beta}\langle\phi_\alpha|\hat{h}|\phi_\beta\rangle + \sum_{a}\sum_{\beta\gamma\delta} C^*_{b\beta}C_{k\gamma}C_{b\delta}\langle\phi_\alpha\phi_\beta||\phi_\gamma\phi_\delta\rangle - \epsilon_k C_{k\alpha} = 0 \ . \tag{4.16}$$

Rewriting the equation we arrive at

$$\sum_{\gamma}\left(\langle\alpha|\hat{h}|\gamma\rangle + \sum_{a}\sum_{\beta\delta} C^*_{b\beta}C_{b\delta}\langle\phi_\alpha\phi_\beta||\phi_\gamma\phi_\delta\rangle\right) C_{k\gamma} = \epsilon_k C_{k\alpha} \ . \tag{4.17}$$

The content within the parenthesis is defined as the Hartree-Fock Hamiltonian:

$$h^{HF}_{\alpha\gamma} = \langle\phi_\alpha|\hat{h}|\phi_\gamma\rangle + \sum_{a}\sum_{\beta\delta} C^*_{b\beta}C_{b\delta}\langle\phi_\alpha\phi_\beta||\phi_\gamma\phi_\delta\rangle \ , \tag{4.18}$$

resulting in the Hartree-Fock equation

$$\sum_{\gamma} h^{HF}_{\alpha\gamma} C_{k\gamma} = \epsilon_k C_{k\alpha} \ . \tag{4.19}$$

The Hartree-Fock equation is nothing but an eigenvalue problem when written in a matrix form

$$\mathbf{h}^{HF}\mathbf{C}_k = \epsilon_k \mathbf{C}_k \ . \tag{4.20}$$

Solving the eigenvalue problem, we find the Hartree-Fock single-particle energies, $\epsilon_k$, and their corresponding eigenvectors. The eigenvectors form the coefficient matrix $\mathbf{C}$. Choosing the eigenstates with the lowest Hartree-Fock energies gives the best choice of orbitals to use in the Hartree-Fock wavefunction. The process of finding the optimal single-particle basis is iterative. We usually start by setting $\mathbf{C}$ equal to unity. Then the Hartree-Fock matrix is constructed and diagonalized. This process is repeated until a convergence criterion is met.

Since most quantum systems cannot be described by a single determinant, the Hartree-Fock method often overshoots the ground state compared with other methods. However, a Hartree-Fock computation is often used to create a single-particle basis used as an input to other many-body methods.

## 4.4 Coupled Cluster

The Coupled Cluster (CC) [3] method aims to solve the time-independent Schrödinger equation

$$\hat{H}\,|\Psi_0\rangle = E_o\,|\Psi_0\rangle \tag{4.21}$$

by expressing the ground state using an exponential ansatz

$$|\Psi_0\rangle = e^{\hat{T}} |\Phi_0\rangle \ , \tag{4.22}$$

where $|\Phi_0\rangle$ is a Slater determinant, and $\hat{T}$ is an excitation operator called the *cluster operator*. The cluster operator is defined as

$$\hat{T} = \hat{T}_1 + \hat{T}_2 + \cdots + \hat{T}_N \ , \tag{4.23}$$

where the sub-indices refers to the number of excitations, so that $\hat{T}_1$ is all single particle excitation, $\hat{T}_2$ all double excitations, etc. We can write these excitation operators explicitly as

$$\hat{T}_1 = \sum_{ai} t_i^a \{a_a^\dagger a_i\} \ , \tag{4.24}$$

$$\hat{T}_2 = \left(\frac{1}{2!}\right)^2 \sum_{abij} t_{ij}^{ab} \{a_a^\dagger a_b^\dagger a_j a_i\} \ , \tag{4.25}$$

$$\vdots \tag{4.26}$$

$$\hat{T}_N = \left(\frac{1}{N!}\right)^2 \sum_{ab\cdots ij\cdots} t_{ij\cdots}^{ab\cdots} \{a_a^\dagger a_b^\dagger \cdots a_j a_i\} \ . \tag{4.27}$$

The exponent in the exponential ansatz is Taylor expanded as

$$e^{\hat{T}} = 1 + \hat{T} + \frac{1}{2!}\hat{T}^2 + \frac{1}{3!}\hat{T}^3 + \cdots \ . \tag{4.28}$$

This sum seems infinite, but it get a natural end if the excitations operator is truncated. The cluster operator is truncated to certain excitations, e.g., in the CCS approximation the cluster operator is truncated at the single excitation level, $\hat{T} \approx \hat{T}_1$, in the CCSD approximation the truncation level is to set to singles and doubles excitations, $\hat{T} \approx \hat{T}_1 + \hat{T}_2$, etc.

The Coupled Cluster method is considered an accurate and numerically tractable method, depending on the truncation level. The use of the cluster operator gives a much better representation of the basis needed to describe electron correlations, compared to, for example, a CI basis.

## 4.5 Quantum Monte Carlo Methods

Quantum Monte-Carlo (QMC) is a family of methods that solves the time-independent Schrödinger equation using a Stochastic approach.

### 4.5.1 Variational Monte Carlo

The basis for Variational Monte Carlo (VMC) [4], is the variational principle. The idea is to approximate the ground state wavefunction with a wavefunction called the *trial wavefunction*, $\psi_T$. A direct application of the variational principle gives us

$$E_0 \leq E_T = \frac{\int_{\mathbf{R}} \psi_T^\dagger(\mathbf{R})\hat{H}(\mathbf{R})\psi_T(\mathbf{R})d\mathbf{R}}{\int_{\mathbf{R}} |\psi_T(\mathbf{R})|^2 d\mathbf{R}} \ ,$$

where $\mathbf{R} = (\mathbf{r}_1, \cdots, \mathbf{r}_N)$ is a vector containing all the coordinate vectors. For an $N$-electron systems, in three-dimensions, the integral is over $3N$ dimensions - making Monte Carlo integration a good choice. Introducing the *local energy*

$$E_L(\mathbf{R}) = \frac{1}{\psi_T(\mathbf{R})}\hat{H}(\mathbf{R})\psi_T(\mathbf{R})$$

we can rewrite $E_T$ as

$$E_T = \int \rho(\mathbf{R})E_L(\mathbf{R})d\mathbf{R} \ ,$$

where $\rho$ is the probability density given by

$$\rho(\mathbf{R}) = \frac{|\psi_T(\mathbf{R})|^2}{\int_{\mathbf{R}} |\psi_T(\mathbf{R})|^2 d\mathbf{R}} \ .$$

The integral is now on the right form for Monte Carlo integration, and by using the Metropolis algorithm [30], the integral is approximated by

$$\langle H \rangle \approx \frac{1}{N}\sum_i E_L(\mathbf{R}_i) \ .$$

By letting each trial wave function have a set of variational parameter $\alpha = (\alpha_1, \cdots, \alpha_m)$, we can calculate

$$\langle H(\alpha) \rangle \approx \frac{1}{N}\sum_i E_L(\mathbf{R}_i, \alpha) \ ,$$

and perform a minimization of the energy with respect to the variational parameters This way we can find the best estimate for the ground state using the trial wavefunction.

### 4.5.2 Diffusion Monte Carlo

Diffusion Monte Carlo (DMC) [5] uses imaginary time propagation to solve the Schrödinger equation for the ground state. If we have a set of energy eigenstates $\{|\psi_i\rangle\}$, the time-independent Schrödinger equation reads

$$\hat{H}\,|\psi_i\rangle = E_i\,|\psi_i\rangle \ .$$

We then 'invent' an operator called the *imaginary time* operator. The imaginary time operator is very similar to the time evolution operator - the difference is that we use $it \to \tau, \tau \in \mathbb{R}$, making it 'imaginary in time'. The action of the imaginary time operator on a state is

$$|\psi(\tau)\rangle = e^{-\hat{H}\tau}\,|\psi(0)\rangle = \sum_i c_i e^{-E_i\tau}\,|\psi_i\rangle \ ,$$

where $\tau$ is the imaginary time parameter. By letting $\tau$ tend towards a large number, the exponential containing the lowest energy will dominate and the higher energy states will fade away. Using the imaginary time operator we can essentially pick out the ground state energy by letting $\tau$ evolve towards a big number:

$$|\psi(\tau \to \infty)\rangle \propto |\psi_0\rangle \ . \tag{4.29}$$

To improve convergence an energy shift, $E_T$, called the *trial energy*, is introduced. Including the trial energy results in

$$e^{-(\hat{H}-E_T)\tau}\,|\psi\rangle = c_0 e^{-(E_0-E_T)\tau}\,|\psi_0\rangle + \sum_{i>0} c_i e^{-(E_i-E_T)\tau}\,|\psi_i\rangle \ .$$

There are now three possible cases:

1. if $E_T < E_0$ then $e^{-(\hat{H}-E_T)\tau}\,|\psi\rangle \to 0$.
2. if $E_T = E_0$ then $e^{-(\hat{H}-E_T)\tau}\,|\psi\rangle \to c_0\,|\psi_0\rangle$.
3. if $E_T > E_0$ then $e^{-(\hat{H}-E_T)\tau}\,|\psi\rangle \to \infty$.

When performing computations a collection of similar computations are performed at the same time. These are referred to as *walkers*. Using a Green's function approach, the Schrödinger equation is rewritten to a diffusion equation, and every time a walker attempts a new move, where the probability of accepting a move is decided by the Green's function. The trial energy is frequently updated during a computation to avoid explosive growth or destruction of walkers. It is essential to start with a good estimate of the ground state and the ground state energy. Usually a VMC computation is performed and the optimal trial wavefunction and trial energy are used as starting point for a DMC calculation.

# Chapter 5

# Time Evolution

In this chapter the formalism of time-development in quantum systems is accounted for, and the most common time evolution methods are described. We will start by studying the time evolution operator. The time evolution operator is the operator that projects the wavefunction from one point in time to another, and it has several interesting properties that we have to take into consideration when we performing numerical computations. Then a quick summary of the most common time evolution methods follows. Some of the methods are directly evolved from their time-independent counterpart, like the Time-Dependent Configuration Interaction (TDCI) [8], Time-Dependent Density Functional Theory (TDDFT) [31], and the Time-Dependent Hartree-Fock (TDHF) [9] method. There are also methods that combine ideas from the time-independent methods mentioned above, creating new methods. Examples are the Multi-Configuration Time-Dependent Hartree (MCTDH) [32], the Multi-Configuration Time-Dependent Hartree-Fock (MCTDHF)[15] and the Orbital Adaptive Time-Dependent Coupled Cluster (OATDCC) [33] method.

## 5.1 The Time Evolution Operator

In quantum mechanics time is not an observable like, for example, the position operator. It is only a parameter, not a measurable quantity. To evolve a state from a time $t_0$ to the time $t$ we introduce the time evolution operator. The time evolution operator, $\hat{U}(t, t_0)$, is a propagator which takes a state from $t_0 \rightarrow t$:

$$|\Psi(t)\rangle = \hat{U}(t, t_0) \, |\Psi(t_0)\rangle \ . \tag{5.1}$$

As we know, a state must be normalized; $\langle\Psi|\Psi\rangle = 1$. Assuming this normalization holds at all times, we find

$$1 = \langle\Psi(t)|\Psi(t)\rangle = \langle\Psi(t_0)|\hat{U}^\dagger\hat{U}|\Psi(t_0)\rangle = \langle\Psi(t_0)|\Psi(t_0)\rangle \ , \tag{5.2}$$

$$\Rightarrow \hat{U}^\dagger\hat{U} = \mathbb{1} \ . \tag{5.3}$$

This means that $\hat{U}^\dagger = \hat{U}^{-1}$ and we can conclude that $\hat{U}$ must be unitary. That the time evolution operator is unitary is an important fact that we have to keep in mind when performing numerical propagation of a wavefunction.

To derive the form of the time evolution operator, substitute $|\Psi(t)\rangle = \hat{U}(t,t_0)\,|\Psi(t_0)\rangle$ in the Schrödinger equation

$$i\frac{\partial U(t,t_0)}{\partial t}\,|\Psi(t_0)\rangle = \hat{H}\hat{U}(t,t_0)\,|\Psi(t_0)\rangle \ . \tag{5.4}$$

This must be true for all states. The relation between the operators are then

$$i\frac{\partial\hat{U}}{\partial t} = \hat{H}\hat{U} \ . \tag{5.5}$$

There are three possible outcomes of this equation depending on the form of the Hamiltonian.

### 1. The time-independent Hamiltonian

For a time-independent Hamiltonian the time evolution operator is found by performing a straightforward integration of Eq. (5.5), ending up with

$$\hat{U}(t) = \exp\left(-i\hat{H}(t-t_0)\right) \ . \tag{5.6}$$

In this case the time evolution operator acting on an eigenstate of the Hamiltonian will just result in a complex phase. This is the form most commonly used in introductory text on quantum mechanics.

### 2. The time-dependent, time-commuting Hamiltonian

If the Hamiltonian commutes with itself at all times, the time evolution operator can be integrated by separating the variables, and the integral is solved the normal way. But, as there is a time-dependence in the Hamiltonian, the integral in the exponent must now include the Hamiltonian. The resulting operator is

$$\hat{U}(t) = \exp\left(-i\int_{t_0}^{t}\hat{H}(t)\ dt\right) \ . \tag{5.7}$$

### 3. The time-dependent, non time-commuting Hamiltonian

If the Hamiltonian does not commute with itself at different times, caution must be exercised when evaluating the integral over the Hamiltonian. The straight forward integration of Eq. (5.5) yields

$$\hat{U}(t) = \mathbb{1} - i \int_{t_0}^{t} \hat{H}(t_1)\hat{U}(t_1)\ dt_1\ , \tag{5.8}$$

where we have used that $\hat{U}(t_0) = \mathbb{1}$. By inserting $\hat{U}$ into itself we get

$$\hat{U}(t) = \mathbb{1} - i \int_{t_0}^{t} \hat{H}(t_1)\ dt_1 + (-i)^2 \int_{t_0}^{t}\int_{t_0}^{t_1} \hat{H}(t_1)\hat{H}(t_2)\hat{U}(t_2)\ dt_1 dt_2\ . \tag{5.9}$$

Performing this recursive iteration infinitely many times results in

$$\hat{U}(t) = \sum_{n=0}^{\infty}(-i)^n \int_{t_0}^{t}\cdots\int_{t_0}^{t_{n-1}} \hat{H}(t_1)\cdots\hat{H}(t_n)\ dt_1\cdots dt_n\ . \tag{5.10}$$

Such a solution is knowns as a *Dyson series*. As the Hamiltonian is not commuting at different times, we must be careful with the ordering. To accomplish such an ordering we introduce the time ordering operator $\mathcal{T}$. The time ordering operator works the following way:

$$\mathcal{T}\left[\hat{H}(t_1)\hat{H}(t_2)\right] = \hat{H}(\{t_1,t_2\}_{max})\hat{H}(\{t_1,t_2\}_{min})\ , \tag{5.11}$$

where $\{t_1,t_2\}_{max}$ picks the larger of $t_1$ and $t_2$, and $\{t_1,t_2\}_{min}$ the smallest. Using the time ordering operator can write Eq (5.10) as

$$\hat{U}(t,t_0) = \sum_{n=0}^{\infty}\frac{(-i)^n}{n!}\mathcal{T}\int_{t_0}^{t}\cdots\int_{t_0}^{t_{n-1}} \hat{H}(t_1)\cdots\hat{H}(t_n)\ dt_1\cdots dt_n \tag{5.12}$$

$$= \mathcal{T}\exp\left(-i\int_{t_0}^{t}\hat{H}(t)\ dt\right)\ . \tag{5.13}$$

Both cases above can be shown as special cases of this form of the time evolution operator.

The standard method for solving the time-dependent Schrödinger equation is to apply the time evolution operator on the wavefunction, propagating it forward in time. Most computational schemes are different ways of approximating the time evolution operator.

## 5.2 A Short Summary of Time-Propagation Methods

In the following subsection a short summary selected time-propagation methods in quantum mechanics is given.

### Time-Dependent Configuration Interaction

The Time-Dependent Configuration interaction (TDCI) [8] method uses a full configuration interaction space as a basis for wavefunction. The scheme is numerically exact given a complete basis. The time-dependent version is very similar to what we have seen in Section 4.2, where the wavefunction is expanded in a Slater determinant basis. Now the same is done, but instead of solving the time-independent Schrödinger equation, we will apply it to the time-dependent Schrödinger equation. The wavefunction is expanded in a Slater determinants basis as

$$|\Psi(t)\rangle = \sum_i C_i(t) \, |\Phi_i\rangle \ , \tag{5.14}$$

where there is now a time-dependence in the expansion coefficients. Using this expansion the Schrödinger equation can be written as a matrix equation

$$i\frac{\partial \mathbf{C}}{\partial t} = \mathbf{H}\mathbf{C} \ , \tag{5.15}$$

where $\mathbf{C}$ is the vector of weights, and $\mathbf{H}$ is the Hamiltonian matrix expressed in the Slater determinant basis. Such a matrix element is

$$H_{ij} = \langle \Phi_i | \hat{H} | \Phi_j \rangle \ . \tag{5.16}$$

The Slater determinant basis can be truncated the same way as with time-independent CI. The computational effort scales exponential with the degrees of freedom, making the method hard to use for larger systems.

### Time-Dependent Hartree-Fock

The Time-Dependent Hartree-Fock [9] method approximates the full wavefunction by a single Slater determinant:

$$|\Psi(t)\rangle = A(t) \, |\phi_1(t) \cdots \phi_N(t)\rangle \ , \tag{5.17}$$

where $A(t)$ is a complex, time-dependent phase and $|\phi_1(t) \cdots \phi_N(t)\rangle$ is an $N$-particle Slater determinant. The Slater determinant is constructed by a set

of orbitals $\{|\phi_i\rangle\}$, representing the single-particle wavefucntions. An orbital is expressed in a time-independent basis, such as

$$|\phi_i(t)\rangle = \sum_\alpha C_{i\alpha}(t)\,|f_\alpha\rangle \ , \tag{5.18}$$

where $|f_\alpha\rangle$ is a time-independent basis function and $C_{i\alpha}(t)$ are expansion coefficients. Using a time-dependent variational principle[1] the equations of motions for $A(t)$ and $\{|\phi_i\rangle\}$ are found. This way the optimal set of orbitals is used.

The TDHF method is a frequently used time evolution method due to its favorable numerical scaling - there is only one configuration. However, the results are often mediocre - one configuration is not enough to describe most systems accurately. For strongly confined systems, one Slater determinant can hold, but for weakly bond system, the correlations between the degrees of freedom becomes more important, which one determinant cannot adequately describe.

There are various modifications of the Time-Dependent Hartree-Fock method attempting to describe the correlations in an acceptable manner.

### Multiconfiguration Time-Dependent Hartree

The Multiconfiguration Time-Dependent Hartree (MCTDH) method was first proposed by Meyer, Manthe and Cederbaum [32] in 1990. In the MCTDH method the wavefunction is written as a linear combination of Hartree products

$$\Psi(q_1,\cdots q_f,t) = \sum_J A_J(t)\Phi_J(q_1,\cdots q_f,t) \ , \tag{5.19}$$

where $f$ are the degrees of freedom, $q$ is a coordinate, $A_J(t)$ is an expansion coefficient and $\Phi_J(q_1,\cdots q_f,t)$ is the Hartree product

$$\Phi_J(q_1,\cdots q_f,t) = \phi_{j_1}(q_1,t)\cdots\phi_{j_f}(q_f,t) \ . \tag{5.20}$$

Notice that there is a time-dependence in the expansion coefficients and the single-particle orbitals. Using the Dirac-Frenkel variational principle, one can find the equations of motion for the expansion coefficient, $A$, to be

$$i\frac{\partial \mathbf{A}}{\partial t} = \mathbf{HA}, \quad J_{IJ} = \langle\Phi_I|\hat{H}|\Phi_J\rangle \ , \tag{5.21}$$

[1]Typically the Dirac-Frenkel variational principle is used in deriving the equations of motion of the time-dependent Hartre-Fock method.

which is basically the same as in TDCI. For the orbitals, the equations of motion are given by

$$i\frac{\partial \boldsymbol{\phi}}{\partial t} = (\mathbb{1} - \mathbf{P})\rho^{-1}\mathcal{H}\boldsymbol{\phi} \ , \tag{5.22}$$

where $\boldsymbol{\rho}$ is the one-body density matrix,$\mathbf{P}$ is the projection operator spanned by the orbital space used, $\boldsymbol{\phi} = (\phi_1, \cdots, \phi_n)^T$ is a vector containing all the orbitals, and $\mathcal{H}$ is the mean-field matrix, where

$$\mathcal{H}_{\gamma\delta} = \langle \Psi_\gamma | \hat{H} | \Psi_\delta \rangle \ , \tag{5.23}$$

where $|\Psi_\gamma\rangle = a_\gamma |\Psi\rangle$, and $a_\gamma$ is an annihilation operator. Note that if the orbital space is complete the $(\mathbb{1} - \mathbf{P})$ term is zero, and we regain the same equations of motion as the standard propagation method. However, if only one configuration is used, it is the same as the time-dependent Hartre-Fock method.

The MCTDHF formulation is general in its form, and systems of non-identical particles can be treated. If the Hamiltonian can be formulated in a product basis, the method is considered very efficient. The method works best for systems with 4-12 degrees of freedom, but for certain problems it can tread higher numbers. The method has been applied to a variety of problems, for example, photodissociation [34], Molecule-surface scattering [35], vibrational predissociation [36].

A general introduction to the MCTDH-method can be found in reference [7], with its corresponding implementation - the Heidelberg package [37], or in the book 'Multidimensional Quantum Dynamics' [16] which also introduces MCTDH for fermions and or bosons.

### Multiconfiguration Time-Dependent Hartree-Fock

The Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF)[15] is the special case of MCTDH for fermionic systems. The equation of motion are the same, the only difference is that the wavefunction expansion coefficients are antisymmetric for the interchange of two particles, thus forming Slater determinants instead of Hartree products. The MCTDHF method can be formulated in its own way, see Chapter 6 for a description and derivation. The formulation of the MCTDHF equations of motion, shown in Chapter 6 allows us to easier take advantage of symmetries, and use the reduced two-body density matrix formalism. For a detailed mathematical analysis of the MCTDHF method, see reference [38].

### Other Mehtods

The Orbital Adaptive Time-Dependent Coupled Cluster (OATDCC) method was proposed by Kvaal [33]. It is a method similar to MCTDH, but instead of using the a full configuration-interaction space it uses the coupled cluster approximation of the configuration-space. For larger systems, where the methods for solving the Schödinger equation directly is too numerically demanding, the Time-Dependent Density Functional theory is the method of choice. For more informations, see 'Fundamentals of Time-Dependent Density Functional Theory' [31].

# Chapter 6

# The Multi-Configuration Time-Dependent HartreeFock Method

The Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) method is a combination of the Time-Dependent Configuration Interaction (TDCI) and the Time-Dependent Hartree-Fock (TDHF) methods. It uses the time-dependent full-configuration interaction space as TDCI does, but in addition places a time-dependence on the single-particle orbitals, as done in TDHF.

## 6.1 Introduction

From a finite set of orthogonal orbitals $\{\phi_i(t)\}$ a full configuration interaction space for an $N$-electron system is constructed. The derivation is based upon the unified view on Multiconfiguration Time-Dependent Hartree-Fock by Alon *et al* [15]. In this space the wavefunction can be approximated by

$$|\Psi(t)\rangle = \sum_n A_n(t)\,|\Phi_n(t)\rangle \ , \tag{6.1}$$

where $|\Phi_n(t)\rangle$ is a Slater determinant and $A_n$ is the weight of the $n$-th Slater determinant. This expansion is exact when the number of orbitals tends towards infinity. A Slater determinant $|\Phi_n(t)\rangle$ is expressed in the Fock space spanned by the orbitals $\{\phi_i(t)\}$ and is written as

$$|\Phi_n(t)\rangle = |\phi_{n_1}(t)\cdots\phi_{n_N}(t)\rangle = |\phi_{n_1}(t)\rangle \otimes \cdots \otimes |\phi_{n_N}(t)\rangle \ . \tag{6.2}$$

Each time-dependent orbital is in turn described in a finite basis with constant basis elements and time-dependent weights, such as

$$|\phi_i(t)\rangle = \sum_{\alpha=1}^{m} C_{\alpha i}(t)\,|\varphi_\alpha\rangle \ , \tag{6.3}$$

where $m$ is the truncation of the space, with the simplest basis being the coordinate space. Having defined the form of the wavefunction we now need to derive the equations of motion for the wavefunction. Instead of using the time-dependent Schrödinger equation to find the dynamics we will take one step back and use the principle of least action to derive the equations of motion. Schrödinger's equation can also be derived from this principle. The action of a system is defined as

$$\mathcal{S}(\{q\}) = \int_{t_0}^{t_1} \mathcal{L}(\{q\})\ dt \ , \tag{6.4}$$

where $\mathcal{S}$ is the action, $\mathcal{L}$ is the Lagrangian, $\{q\}$ is the set of generalized coordinates needed to describe the system uniquely, and the integral is defined over the time $t_0$ to $t_1$. The assumption is that the equations of motions are found when the action is stationary, meaning when $\delta S = 0$. For our system we have certain restrictions, such as the orthogonality of the orbitals. Such restrictions on the system are introduced in the form of Lagrangian multipliers. For a quantum system the Lagrangian is

$$\mathcal{L} = \langle\Psi|\hat{H} - i\frac{\partial}{\partial t}|\Psi\rangle \ , \tag{6.5}$$

and the Lagrangian multipliers are

$$-\sum_{pq} \lambda_{pq}(\langle\phi_p|\phi_q\rangle - \delta_{pq}) = 0 \ . \tag{6.6}$$

For the basis we have chosen, the generalized coordinates are $\{A_n\}, \{A_n^\dagger\}, \{\langle\phi_i|\}$ and $\{|\phi_i\rangle\}$. Inserting these expressions gives us the action

$$\mathcal{S} = \int \left[\langle\Psi|\hat{H} - i\frac{\partial}{\partial t}|\Psi\rangle - \sum_{pq} \lambda_{pq}(\langle\phi_p|\phi_q\rangle - \delta_{pq})\right] dt \ . \tag{6.7}$$

In order to derive the equations of motion we need to find the stationary solutions of the action by varying the coefficients of the Slater determinants $\{A_n\}, \{A_n^\dagger\}$ and the orbitals $\{\langle\phi_i|\}$ and $\{|\phi_i\rangle\}$. Using the reduced density

matrices[1] the Hamiltonian can be written explicitly in terms of orbitals. The reduced density matrices for one and two particles are

$$\rho_{pq} = \langle \Psi | a_p^\dagger a_q | \Psi \rangle \tag{6.8}$$

$$\rho_{pqrs} = \langle \Psi | a_p^\dagger a_q^\dagger a_r a_s | \Psi \rangle \ , \tag{6.9}$$

and they are independent of the form of the orbitals. Using second quantization and the reduced density matrix elements, the expectation value of the Hamiltonian is

$$\langle \Psi | \hat{H} | \Psi \rangle = \sum_{pq} \rho_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle + \frac{1}{2} \sum_{pqrs} \rho_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_s \phi_r \rangle \ . \tag{6.10}$$

The time-derivative operator applied to the full wavefunction is

$$\langle \Psi | \frac{\partial}{\partial t} | \Psi \rangle = \sum_{pq} \rho_{pq} \langle \phi_p | \frac{\partial \phi_q}{\partial t} \rangle + \sum_n A_n^\dagger \frac{\partial A_n}{\partial t} \ , \tag{6.11}$$

where we have used that the time-derivative can be written as a single-particle operator when acting on a Slater determinant

$$\frac{\partial}{\partial t} | \Phi_n \rangle = \sum_{pq} \langle \phi_p | \frac{\partial \phi_q}{\partial t} \rangle a_p^\dagger a_q \, | \Phi_n \rangle \ . \tag{6.12}$$

Combining all the terms gives the following Lagrangian

$$\mathcal{L} = \sum_{pq} \rho_{pq} \left( \langle \phi_p | \hat{h} | \phi_q \rangle - i \langle \phi_p | \frac{\partial \phi_q}{\partial t} \rangle \right) \tag{6.13}$$

$$+ \frac{1}{2} \sum_{pqrs} \rho_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_s \phi_r \rangle - i \sum_n A_n^\dagger \frac{\partial A_n}{\partial t} \ . \tag{6.14}$$

We now have everything we need to find the equation of motion. Varying the action $\mathcal{S}$ with respect to all generalized coordinates results in

$$\delta S = \int \sum_i \left[ \frac{\partial}{\partial q_i} \left( \mathcal{L} - \sum_{pq} \lambda_{pq} ( \langle \phi_p | \phi_q \rangle - \delta_{pq} ) \right) \delta q_i \right] dt = 0 \ , \tag{6.15}$$

where $q_i$ is a degree of freedom. Since $\delta q_i$ is an arbitrary variation it follows that

$$\frac{\partial}{\partial q_i} \left( \mathcal{L} - \sum_{pq} \lambda_{pq} ( \langle \phi_p | \phi_q \rangle - \delta_{pq} ) \right) = 0 \ . \tag{6.16}$$

By performing variations of all $q_i \in [\{\langle \phi_j |\}, \{|\phi_j\rangle\}, \{A_n\}, \{A_n^\dagger\}]$ we can find their respective equations of motion.

[1] See section 3.4 for more details on density operators.

## 6.2 Orbital Equations of Motion

To find the equations of motion for an orbital $|\phi_i\rangle$ we calculate Eq. (6.16) with $q_i = \langle\phi_i|$. Taking the derivative results in

$$\sum_q \rho_{iq} \left( \hat{h} \left|\phi_q\right\rangle - i \left| \frac{\partial \phi_q}{\partial t} \right\rangle \right) + \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle - \sum_q \lambda_{iq} \left|\phi_q\right\rangle = 0 \ . \tag{6.17}$$

Solving for the Lagrangian multipliers, we obtain

$$\sum_q \lambda_{iq} \left|\phi_q\right\rangle = \sum_q \rho_{iq} \left( \hat{h} \left|\phi_q\right\rangle - i \left| \frac{\partial \phi_q}{\partial t} \right\rangle \right) + \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle \ , \tag{6.18}$$

where we have used that

$$\frac{\partial}{\partial \left\langle \phi_i \right|} \sum_{pqrs} \rho_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_s \phi_r \rangle = \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle + \sum_{prs} \rho_{pirs} \langle \phi_p | \hat{V} | \phi_s \rangle \otimes |\phi_r\rangle \tag{6.19}$$

$$= 2 \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle \ . \tag{6.20}$$

by exploiting the symmetries of $\rho_{pqrs}$. Multiplying Eq. (6.17) with $\langle\phi_j|$ gives an explicit expression for the Lagrangian multipliers:

$$\lambda_{ij} = \left\langle \phi_j \right| \left[ \sum_q \rho_{iq} \left( \hat{h} \left|\phi_q\right\rangle - i \left| \frac{\partial \phi_q}{\partial t} \right\rangle \right) + \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle \right] \ . \tag{6.21}$$

We can now write the Lagrangian multiplier term in Eq. (6.17) as

$$\sum_j \lambda_{ij} \left|\phi_j\right\rangle = \hat{P} \left[ \sum_q \rho_{iq} \left( \hat{h} \left|\phi_q\right\rangle - i \left| \frac{\partial \phi_q}{\partial t} \right\rangle \right) + \sum_{qrs} \rho_{iqsr} \langle \phi_q | \hat{V} | \phi_s \rangle \otimes |\phi_r\rangle \right] \ , \tag{6.22}$$

where $\hat{P}$ is the projection operator defined as

$$\hat{P} = \sum_{j=1}^{n} \left|\phi_j\right\rangle \left\langle \phi_j \right| \ , \tag{6.23}$$

where $n$ is the truncation of the orbitals. Eq. (6.17) can now be rewritten as

$$(\mathbb{1} - \hat{P}) \left[ \sum_q \rho_{iq} \left( \hat{h} \left|\phi_q\right\rangle - i \left| \frac{\partial \phi_q}{\partial t} \right\rangle \right) + \sum_{qrs} \rho_{iqrs} \langle \phi_q | \hat{V} | \phi_r \rangle \otimes |\phi_s\rangle \right] = 0 \ . \tag{6.24}$$

Solving for the time derivative of the orbitals:

$$(\mathbb{1}-\hat{P})i\sum_q \rho_{iq}\left|\frac{\partial\phi_q}{\partial t}\right\rangle = (\mathbb{1}-\hat{P})\left[\sum_q \rho_{iq}\hat{h}\,|\phi_q\rangle + \sum_{qrs}\rho_{iqrs}\langle\phi_q|\hat{V}|\phi_r\rangle\otimes|\phi_s\rangle\right] . \tag{6.25}$$

To solve for one of the orbitals we exploit the properties of the inverse one-body density matrix by multiplying by $\sum_i \rho^{-1}_{ji}$, giving us

$$(\mathbb{1}-\hat{P})i\left|\frac{\partial\phi_j}{\partial t}\right\rangle = (\mathbb{1}-\hat{P})\left[\hat{h}\,|\phi_j\rangle + \sum_{iqrs}\rho^{-1}_{ji}\rho_{iqrs}\langle\phi_q|\hat{V}|\phi_r\rangle\otimes|\phi_s\rangle\right] . \tag{6.26}$$

To simplify the notation we redefine the mean field operator $\langle\phi_q|\hat{V}|\phi_r\rangle$ to $\hat{V}^{qr}$, resulting in the working equation

$$i(\mathbb{1}-\hat{P})\left|\frac{\partial\phi_j}{\partial t}\right\rangle = (\mathbb{1}-\hat{P})\left[\hat{h}\,|\phi_j\rangle + \sum_{iqrs}\rho^{-1}_{ji}\rho_{iqsr}\hat{V}^{qr}\,|\phi_s\rangle\right] . \tag{6.27}$$

For the solution to be unique we also have to introduce a constraining operator

$$i\langle\phi_i|\frac{\partial\phi_j}{\partial t}\rangle = \langle\phi_i|\hat{g}|\phi_j\rangle = g_{ij} . \tag{6.28}$$

Note that $g_{ij}=0$ is a valid solution[2]. Using this constraint, the equations of motion for the orbitals are

$$i\left|\frac{\partial\phi_j}{\partial t}\right\rangle = \sum_j g_{ij}\,|\phi_j\rangle + (\mathbb{1}-\hat{P})\left[\hat{h}\,|\phi_j\rangle + \sum_{iqrs}\rho^{-1}_{ji}\rho_{iqrs}\hat{V}^{qr}\,|\phi_s\rangle\right] . \tag{6.29}$$

### 6.2.1 Spatial Discretization of the orbital equations

Taking the projection of Eq. (6.29) (with $g_{ij}=0$) onto position space results in

$$i\frac{\partial\phi_j(\mathbf{r})}{\partial t} = (\mathbb{1}-\hat{P})\left[\hat{h}\phi_j(\mathbf{r}) + \sum_{iqrs}\rho^{-1}_{ji}\rho_{iqrs}\hat{V}^{qr}(\mathbf{r})\phi_s(\mathbf{r})\right] . \tag{6.30}$$

Discretizing the orbitals on a spatial grid,

$$\mathbf{c}_j = [\phi_j(\mathbf{r}_0),\cdots,\phi_j(\mathbf{r}_N)]^T , \tag{6.31}$$

[2]One that has been used throughout this thesis.

gives us the equation

$$i\frac{\partial \mathbf{c}_j}{\partial t} = (\mathbb{1} - \mathbf{P})\left[\mathbf{h}\mathbf{c}_j + \sum_{iqrs} \rho_{ji}^{-1}\rho_{iqrs}\mathbf{V}^{qr}\mathbf{c}_s\right] \ . \tag{6.32}$$

We redefine the last part as

$$\sum_i \rho_{ji}^{-1} \sum_{qrs} \rho_{iqrs}\mathbf{V}^{qr}\mathbf{c}_s = \sum_i \rho_{ji}^{-1}\mathbf{O}_i \ , \tag{6.33}$$

where

$$\mathbf{O}_i = \sum_{qrs} \rho_{iqrs}\mathbf{V}^{qr}\mathbf{c}_s \ . \tag{6.34}$$

The equations of motion for all the orbitals can now be summarized as a matrix equation

$$i\frac{\partial \mathbf{C}}{\partial t} = (\mathbb{1} - \mathbf{P})\left[\mathbf{h}\mathbf{C} + \boldsymbol{\rho}^{-1}\mathbf{O}\right] \ , \tag{6.35}$$

with

$$\mathbf{C} = [\mathbf{c}_1, \cdots \mathbf{c}_n] \ . \tag{6.36}$$

## 6.3 Equations of Motion for the Wavefunction Expansion Coefficients

We will now find the equations of motion for the expansion coefficients, $\{A_n\}$, of the wavefunction, by taking the derivative of the action, as described in Eq. (6.7), with respect to $A_i$. This results in

$$\langle \Phi_i | \hat{H} | \Psi \rangle - i \langle \Phi_i | \frac{\partial \Psi}{\partial t} \rangle = 0 \ , \tag{6.37}$$

where we have used that $\frac{\partial |\Psi\rangle}{\partial A_i} = |\Phi_i\rangle$. Expanding the wavefunction by using Eq. (6.1) and performing some reorganizing gives us

$$i\frac{\partial A_i}{\partial t} = \sum_j \langle \Phi_i | \hat{H} - i\frac{\partial}{\partial t} | \Phi_j \rangle \ , \tag{6.38}$$

which can straightforwardly be written as a matrix vector equation

$$i\frac{\partial \mathbf{A}}{\partial t} = \boldsymbol{\mathcal{H}}\mathbf{A} \ , \tag{6.39}$$

with

$$\mathcal{H}_{ij} = \langle \Phi_i | \hat{H} - i\frac{\partial}{\partial t} | \Phi_j \rangle \ , \quad \mathbf{A} = \begin{bmatrix} A_1 \\ A_2 \\ \vdots \\ A_M \end{bmatrix} \ . \tag{6.40}$$

Let us simplify this by setting the constraining operator, $g_{ij}$, to zero. The Hamilton matrix elements are then

$$\mathcal{H}_{ij} = \langle \Phi_i | \hat{H} | \Phi_j \rangle \ . \tag{6.41}$$

The Hamiltonian matrix is set up using the Hamiltonian operator in second quantization

$$\hat{H} = \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle a_p^\dagger a_q + \frac{1}{2} \sum_{pqrs} \langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle a_p^\dagger a_q^\dagger a_s a_r \ . \tag{6.42}$$

## 6.4 The Constraint Operator

In the previous section we introduced the constraining operator, $\hat{g}$, to lift the ambiguity of both the orbitals and the wavefunction coefficients having a time-dependence. The ansatz for the wavefunction, given in Eq. (6.1), is not unique when there is a time dependence in both the expansion coefficients and the orbitals. To counter this, we introduce an additional constraint on the orbitals:

$$i\langle \phi_i | \frac{\partial \phi_j}{\partial t} \rangle = \langle \phi_i | \hat{g} | \phi_j \rangle = g_{ij} \ , \tag{6.43}$$

where $\mathbf{g}$ is an arbitrary time-dependent Hermitian matrix. The constraining operator can be chosen arbitrary - the accuracy of the solution is not affected [32]. There are, however, impacts on the numerical performance depending on the form of the constraining operator. When choosing the form of the constraining operator there are two obvious choices: $\hat{g} = 0$ and $\hat{g} = \hat{h}$. Choosing the latter results in the equations of motion for the Slater determinants to somewhat simplified. The Hamiltonian only has the *residual* terms - i.e., only the interaction part of the Hamiltonian matrix needs to be constructed. The choice of constraining operator is in a way choosing whether to put the workload on the orbitals or on the Slater determinants [32]. All computations done in this thesis use $\hat{g} = 0$.

# Chapter 7

# The Inital State

Before we can perform any time evolution in a system, we need to prepare an initial state. There are several ways of creating an initial state. The method we will use, is an iterative scheme called *imaginary time propagation* or *energy relaxation*. We can use imaginary time propagation to find the ground state of a system, and use the ground state as a starting point for time-dependent studies.

## 7.1 Imaginary Time Propagation

Using imaginary time propagation, we can rewrite the MCTDHF equations of motion to find the ground state of a system. Using an imaginary time $t = -i\tau$, with $\tau \in \mathbb{R}$, the time evolution operator changes from $\exp\left(-i\hat{H}t\right)$ to $\exp\left(-\hat{H}\tau\right)$. Applying this new imaginary time operator on an arbitrary state, $|\Psi(\tau)\rangle$, yields

$$|\Psi(\tau)\rangle = e^{-\hat{H}\tau}\,|\Psi(0)\rangle = \sum_i c_i e^{-E_i\tau}\,|i\rangle \ , \tag{7.1}$$

where $\{|i\rangle\}$ are the energy eigenstates of the Hamiltonian. Letting $\tau$ tend towards a large number the smallest eigenvalue will dominate, resulting in

$$|\Psi(\tau \to \infty)\rangle \to c_0 e^{-E_0\tau}\,|0\rangle \ , \tag{7.2}$$

and we find the ground state. We can therefore use the MCTDHF equations of motion to propagate out the ground state by exchanging $t = -i\tau$. This gives us

$$\frac{\partial \mathbf{C}}{\partial \tau} = -\left(\mathbb{1} - \mathbf{P}\right)\left[\mathbf{h}\mathbf{C} + \boldsymbol{\rho}^{-1}\mathbf{O}\right] \ , \tag{7.3}$$

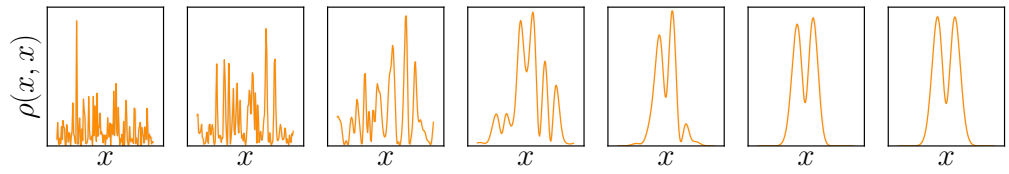



Figure 7.1: Example of imaginary time propagation for a system where the initial orbitals are initially set by a random unitary matrix. The plot shows how the one-body density evolves as a function of imaginary time steps, ending in the ground state density.

and

$$\frac{\partial \mathbf{A}}{\partial \tau} = -\boldsymbol{\mathcal{H}}\mathbf{A} \ . \tag{7.4}$$

The initial wavefunction is chosen arbitrarily. During imaginary time propagation the $\mathbf{A}$ vector must be re-orthonormalized due to the loss of the norm. The scheme can be modified so that the orthonormalization is not needed [7] by writing the equation for the wavefunction expansion coefficients as

$$\frac{\partial \mathbf{A}}{\partial \tau} = -(\boldsymbol{\mathcal{H}} - \mathbb{1}E)\mathbf{A} \ , \tag{7.5}$$

where

$$E = \frac{\mathbf{A}^\dagger \boldsymbol{\mathcal{H}} \mathbf{A}}{\mathbf{A}^\dagger \mathbf{A}} \ . \tag{7.6}$$

For the imaginary time propagation scheme to work the initial wavefunction must have an adequate overlap with the ground state. If the separation between the ground state and the first excited state is very small, the convergence will be very slow.

Figure 7.1 shows an example of imaginary time propagation. The figure shows how the one-body density of a system gradually evolves with the number of imaginary time-steps taken. The initial orbitals and wavefunction expansion coefficients are chosen at random. Convergence is clearly shown in the figure as the density evolves towards a stable solution.

# Part II

# Implementation and Validation

# Chapter 8

# Implementation

In this chapter the numerical implementation of the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) method is presented in detail. For the theory behind the MCTDHF method, see Chapter 6. We start by describing the basic structure of the overall implementation, and afterwards present specific details.

All the numerically demanding parts are written using C++. Code development is done using QtCreator[1], and the project is best viewed from Github or by importing the Qt project file into QtCreator. Data-analysis is performed using several small Python scripts. The MCTDHF program developed as a part of this thesis is made freely available at `https://github.com/sigvebs/MCTDHF` under a GPL license.

### Object-Orientation and Code Structure

The code is always kept as general as possible, using an object-oriented approach to programming. This makes it easy to change options in the code and implement new features. Typically we want to test different operators, integrators, bases, etc. The whole program is written by wrapping all functionality into classes, and whenever possible use abstract base classes.

The source code is found in the 'src'-folder. All classes have their own folder, and for abstract classes there is a sub-folder named 'implementation' where specific implementation is found.

[1] QtCreator is available at `http://qt-project.org/` under a GPL v3 license, or at `http://qt.digia.com/` under a commercial license.

## 8.1 Overall Structure of the MCTDHF Code

We want to solve the MCTDHF Equations of Motion (EOM) for the orbitals and the expansion coefficients. That is, the coupled set of non-linear differential equations given by Eq. (6.39) and Eq. (6.29). But before we can begin solving the EOMs the system must be defined and initialized. The overall structure of the MCTDHF program is split into three steps:

1. Initialization of the system.

2. Imaginary time propagation of the wavefunction.

3. Real time propagation of the wavefunction.

Figure 8.1 shows the basic flow of the program, which initializes the system and performs imaginary time and/or time propagation. After the MCTDHF computations are complete, Python scrips are used to analyze the results. We will now give a short summary of these three steps.

### Initialization

The first five steps of Figure 8.1 shows the initialization process. To begin, the compiled executable must be supplied with a configuration file[2]. The configuration file contains all information needed for a computation and is loaded into a configuration object in the program. This configuration object is shared with all class objects in the program. Based on the information in the configuration file the following is done:

1. The spatial grid is constructed, either by loading the discretization from file or creating a new one.

2. All the single particle quantum numbers are created within a closed shell. These represent the orbitals. The orbitals are discretized on the spatial grid, or loaded from file. The resulting discretization is stored in a complex matrix.

3. The configuration space is constructed by creating a Slater determinant for every possible $N$-particle configuration made form the set of single-particle orbitals. The Slater determinants are stored in a binary form using the occupation number representation.

[2] The C++ library 'libconfig' [39] is used to read, manipulate, and write structured configuration files using the cfg-file format.

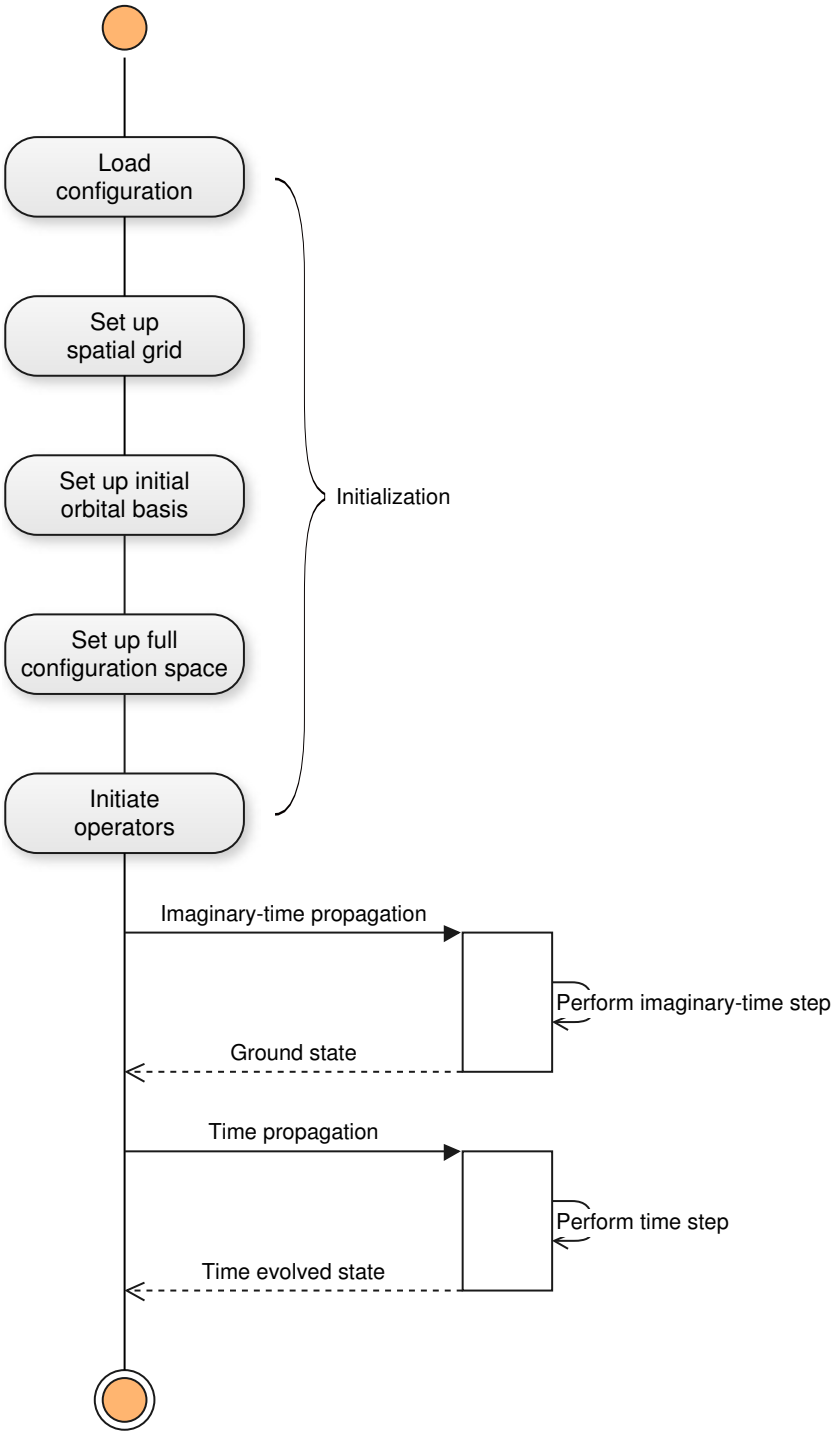



Figure 8.1: The basic flow of the MCTDHF program. The program starts by initializing the system. After the initialization either imaginary time propagation and/or time propagation is performed.

4. The one-body operators, the interaction operators, and the differential operators are defined and initialized.

Details of the steps above are given in the subsequent sections. Now the program is ready for imaginary time propagation and/or an ordinary time propagation.

### Imaginary Time Propagation

Imaginary time propagation[3] is used to find the ground state wavefunction. For imaginary time propagation Eq. (7.5) is used instead of Eq. (6.39) for the EOM of the wavefunction's expansion coefficients. To solve the EOMs for imaginary time we must first choose an integration scheme[4]. At every time step the right hand side of the EOMs must be computed. To that we must first:

1. Compute the mean fields.
2. Recalculate every one- and two-body matrix elements.
3. Reconstruct the entire Hamiltonian matrix.
4. Calculate the reduced one- and two-body matrices.
5. Compute the projection operator.

The right hand side can now be calculated. The integration is performed, and the right hand side of the EOMs are computed according to Section 8.3. The integration is continued until a convergence criterion is met and/or a set number of integration steps are performed.

### Time Propagation

The MCTDHF EOMs are integrated forward in time. The process is almost identical to the imaginary time propagation, except for the EOM of the wavefunction's expansion coefficients, which is now given by Eq. (6.39). The initial state of the wavefunction is given by either loading an state from file, or performing an imaginary time propagation so that we are starting in the ground state. The propagation continues either until a specific time is met or a predetermined number of steps are performed.

[3] The concept of imaginary time propagation is described in Chapter 7.

[4] The different choices of integrators, how they work, and their implementation are described in Chapter 10.

## 8.2 A description of the Basic Classes / Functionality

In the following subsections we discuss the implementation of the spatial grid, the orbitals, the Slater determinants, potentials and differential operator, interaction elements, and some other operations like the implementation of second quantization operators.

### 8.2.1 Grid Representation

The spatial discretization of the grid is stored in the `grid` -class. Currently only a finite, uniform grid is implemented, but the code is created in such a way that other bases can easily be implemented. First the grid must be initialized, either by creating a new discretization or loading one from file. A grid is specified by a range, grid spacing, boundary conditions, and the number of dimensions. The discretized grid is stored in a matrix, $\mathbf{R} \in \mathbb{R}^{d \times N_{grid}}$, where $d$ is the number of spatial dimensions and $N_{grid}$ is the number of grid vectors. Every column in the $\mathbf{R}$-matrix contains a vector with $d$-coordinates: If another object needs a grid-vector it must call the `grid` -objects function `at( int i )`, and a reference to the $i$-th column vector in $\mathbf{R}$ is returned. Most objects in the MCTDHF application store a reference to the `grid` for easy access to the discretization. The implementation is currently only supporting Cartesian coordinates on an uniform grid, but as the program has been made to be modular, adding other type of grids should not be too much additional work.

### 8.2.2 Orbitals

The orbitals are the single-particle wavefunctions. An orbital is defined by a set of quantum numbers, e.g., spin, angular momentum, etc. There are, however, an infinite set of such orbitals. In numerical calculations we most impose a truncation of the orbitals space.

#### The Single Particle Shell Model

As an example we will use the Harmonic oscillator basis (in two-dimensions and Cartesian coordinates) to show how an orbital space is chosen. The energy of a single harmonic oscillator function is

$$\epsilon_{n_x n_y} = \omega(1/2 + n_x + n_y), \tag{8.1}$$

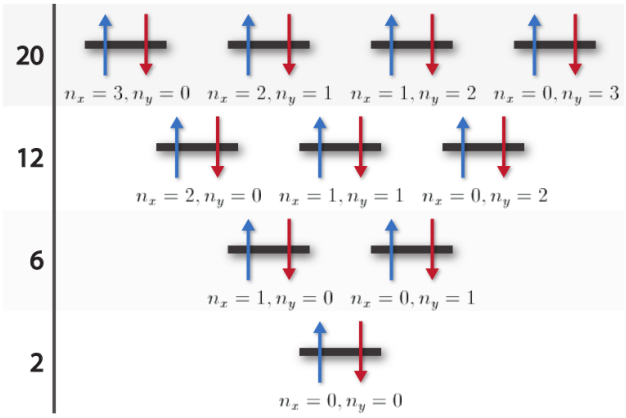



Figure 8.2: The shell structure of a 2D quantum harmonic oscillator for the electron. The numbers on the y-axis are the so called magic numbers forming closed shells. Every level exhibits a double degeneracy due to the electron spin. See the text for an explanation of the quantum numbers.

where $\omega$ is the oscillator frequency and $n_x$ and $n_y$ are Cartesian quantum numbers. Figure 8.2 shows the possible quantum numbers for electrons. The arrows represent the electron spin. The horizontal lines show degenerate energy shells where the orbitals have the same energy, such as the two first orbitals ($n_x = 0, n_y = 0$) form the first shell $R = 1$, the next shell, $R = 2$, contains four orbitals, and $R = 3$ contains six orbitals. To create our single particle basis, a truncation is performed by specifying a maximal shell number. All orbitals within this shell and below are included in our *model space*, forming a set of orbitals $\{\phi_i\}$. The quantum numbers of all the orbitals are stored in an STL[5] vector called `states`. After every orbital in $\{\phi_i\}$ has been defined, a spatial discretization of the orbitals is performed.

### Spatial Discretization

The spatial representation of the orbitals is stored in the matrix $\mathbf{C} \in \mathbb{C}^{N_{grid} \times N_{orb}/2}$, where $N_{grid}$ is the number of grid vectors and $N_{orb}$ is the total number of orbitals[6]. A column in $\mathbf{C}$ represents a discretized orbital. The discretization is stored in the same order as the coordinate vectors in the $\mathbf{R}$-matrix from the

[5] The Standard Template Library, or STL, is a C++ library of container classes, algorithms, and iterators [40].

[6] Since we are only simulating electrons, and not using magnetic fields, only half of the orbitals needs to be discretized in $\mathbf{C}$, due to the spatial representations of spin up and down orbitals being the same.

grid -class, such that

$$\phi_\alpha(\mathbf{r}_j) = C_{j\alpha} \ , \tag{8.2}$$

where $\mathbf{r}_j$ is the $j$-column of $\mathbf{R}$. Whenever other parts of the program needs access to the orbitals, a constant reference to $\mathbf{C}$ is used. Currently the program supports an initial discretization in one of the forms: harmonic oscillators, hydrogen-like, or a random, unitary matrix. The initial discretization can also be loaded from file.

#### Discrete Variable Representation

The discrete variable representation has not been used in this thesis due to pressing time. It is, however, the method of choice if we want to compute systems containing more particles, and the hope is to implement it the future. For an introduction to the Discrete Variable Representation the reader is directed to reference [41]. For applications of DVRs in the MCTDH-method, see reference [16].

### 8.2.3 Slater determinants

The full many-body wavefunction is written as a linear combination of Slater determinants[7]:

$$|\Psi\rangle = \sum_n A_n \, |\Phi_n\rangle \ . \tag{8.3}$$

This relation is exact if the complete set of all $N$-particle Slater determinants are used. For numerical purposes we have to impose a truncation on the space used. We will use the *full configuration interaction*-space. In the full configuration space all possible $N$-particle Slater determinants are generated from a set of orbitals $\{\phi_i\}$. The number of Slater determinants used to represent the full wavefunction is therefore given by the binomial factor $\binom{N_{orb}}{N}$, where $N_{orb}$ is the total number of orbitals. In practice, we use the *odometer*-algorithm [42] to generate all possible $N$-particle Slater determinants from $\{\phi_i\}$. We can also introduce other restrictions on the configuration space, like only allowing Slater determinants with a specific total spin value - this is called the *M-scheme* [29]. Whenever a new Slater determinant is created it is stored in a binary form using the *occupation number representation*.

#### Occupation Number Representation

The occupation number representation is a way to describe a Slater determinant is using a string of numbers that are either zero or one. If position

[7] See Chapter 3 for more details.

$i$ is set to one, it means that the $i$-th orbital is occupied. An example for a 3-particle Slater determinant:

$$|\Phi_{0,2,9}\rangle = |\phi_0\phi_2\phi_9\rangle = |10100000010\cdots\rangle \ . \tag{8.4}$$

Such a representation is very convenient when working with computers due to the binary representation of numbers on computer. To store a Slater determinant in the occupation number representation, a number of bits[8] is dedicated to storing a Slater determinant. On a computer the smallest configuration of bits is a *byte* - comprised of eight bits. For optimal efficiency we want to keep the number of bits in numbers of 32 or 64, depending on the architecture of the computer used. Most modern computers are capable of 64-bit processing. For configurations not exceeding the capabilities of a 64-bit representation, a fundamental data type like the integer can be used. For the program to handle larger numbers of configurations we will instead use the STL bitset library, where one can dedicate any number of bits. Listing 8.1 shows an example of how the bitset library can be used to create the Slater determinant described in Eq. (8.4).

```
#DEINFE BITS 10
#include <bitset>

bitset<BITS> binState;

binState.set(0)
binState.set(2)
binState.set(9)

cout << binState << endl;
// The results of the print statement is
// 1010000001
```

Listing 8.1: Example use of the bitset library to create a Slater determinant using the occupation number representation

### 8.2.4 Creation and Annihilation Operators

Creation and annihilation operators[9] hold an important role in the code. To implement the creation and annihilation operators when a Slater determinant is expressed using the occupation number representation is, in fact, quite

[8] A bit is the smallest unit of information on a computer and it can take the value of either zero or one.

[9] See Section 3.3 for more details on Creation and annihilation operators.

easy. The creation and annihilation operators work on a Slater determinant the following way

$$a_i \left|\Phi_{a,b,\cdots,i,\ldots}\right\rangle = (-1)^{n_p} \left|\Phi_{a,b,\ldots}\right\rangle \ , \quad a_i \left|\Phi_{a,b,\ldots}\right\rangle = 0 \quad \text{for } i \notin a,b,\cdots \ , \tag{8.5}$$

$$a_i^\dagger \left|\Phi_{a,b,\ldots}\right\rangle = (-1)^{n_p} \left|\Phi_{a,b,\cdots,i,\ldots}\right\rangle \ , \quad a_i^\dagger \left|\Phi_{a,b,\cdots,i,\ldots}\right\rangle = 0 \ , \tag{8.6}$$

where $n_p$ is the number of filled states before $i$. To implement the creation and annihilation operators we need a set of functions that can add or remove an orbital from a Slater determinant, and calculates the sign. Listing 8.2, Listing 8.3 and Listing 8.4 shows the code used to implement the basic operations used for creation and annihilation operators. To represent the null-state the last bit is set to one. The null-state is used to check whether to performer calculations on a Slater determinant or not. This way we can save computational resources.

```
inline int sign(const int n, const bitset<BITS> &state)
{
  int s = 1;
  for (int i = 0; i < n; i++) {
    if (state[i] != 0) {
      s *= -1;
    }
  }
  return s;
}
```

Listing 8.2: Implementation of the sign-function used in creation and annihilation operators. The $n$ parameter tells us which orbital the orbitals some action was performed on. The *state* parameter is a Slater determinant represented in a binary form.

```
inline void addParticle(const int n, bitset<BITS> &state)
{
  bitset<BITS> a;
  a.set(n);

  // & is a binary comparison of two numbers
  bitset<BITS> comp = a & state;

  if (comp.count() == false)
    state.set(n);
  else
    state.set(BITS - 1);
}
```

Listing 8.3: Implementation of the addParticle-function used in creation operators.

```
inline void removeParticle(const int n, bitset<BITS> &
    state)
{
  bitset<BITS> a;
  a.set(n);

  // & is a binary comparison of two numbers
  bitset<BITS> comp = a & state;

  if (comp.count() == true)
    state.set(n, 0);
  else
    state.set(BITS - 1);
}
```

Listing 8.4: Implementation of the removeParticle-function used in annihilation operators.

### 8.2.5 One-Body Operators/Potentials

The one-body operators are implemented through abstracts classes to uphold the generality of the code. For the purpose of implementation we have separated the implementation of differential operators and potentials. Specific operators/potentials are implemented through a subclass of the abstract base class. To include a new potential in the code a subclass of `potential` must be created in 'src/Potential/implementation' and an identifier must be added in the definition file, found in 'src/includes/defines.h'. The differential operator and a list of potentials must be specified in the configuration file. See Listing 8.5 for an illustrative example of the implementation the harmonic oscillator potential. All potentials must implement a function `evaluate(const cx_vec &psi, double t)`. This function evaluates the action of the potential on an discretized orbital $\phi$, where $\phi$ is given by `const cx_vec &phi`. The time parameter, $t$, does not need to be specified for time-independent potentials. Potentials are diagonal in position space, thus the discretization of the potential can be represented by a vector. The implementation of differential operators is very similar.

```
HarmonicOscillator::HarmonicOscillator(Config *cfg, const
    Grid &grid):Potential(cfg, grid)
```

```
{
  // Reading variables from the configuration object
  ...

  potential = vec(nGrid);

  // Constructing the discretized version of the harmonic
     oscilaltor potential.
  for(int j=0; j<nGrid; j++){
    const vec& r = grid.at(j);
    double r2 = 0;
    for(int i=0; i<dim; i++){
      r2 += r(i)*r(i);
    }
    potential(j) = 0.5*w*w*r2;
  }
}

cx_vec HarmonicOscillator::evaluate(const cx_vec &phi,
    double t)
{
    return potential % phi;
}
```

Listing 8.5: Example implementation of the Harmonic oscillator potential.

More information on the differential operators and their implementation is found in Chapter 9. For the potentials used in this thesis, see Chapter 13.

All the one-body potentials and the differential operators are stored in a general class `SingleParticleOperator`. There, all general operations regarding one-body operations are implemented, like the potential acting on all the orbitals in position space, the computation of the second derivative of the orbitals, and the computation of the one-body Hamiltonian matrix. If asked it can return references to the one-body Hamiltonian matrix or the matrix containing the action one-body Hamiltonian operators on all orbitals (in position space, used in the EOM for the orbitals).

### 8.2.6 Two-Body Operators / Interaction Elements

The interaction elements are given by

$$V_{pqrs} = \langle \phi_p \phi_q | \hat{V}_I | \phi_r \phi_s \rangle \ , \tag{8.7}$$

where $p, q, r$ and $s$ are indices of orbitals and $\hat{V}_I$ is the interaction operator. The interaction elements are computed by performing integrals over the mean

fields[10]. From a mean field, $V^{qs}(\mathbf{r})$, we can calculate an interaction element by

$$V_{pqrs} = \langle \phi_p | \hat{V}^{qs} | \phi_r \rangle = \int \phi_p^\dagger(\mathbf{r}) V^{qs}(\mathbf{r}) \phi_r(\mathbf{r}) \; d\mathbf{r} \; . \tag{8.8}$$

These integrals are typically solved using the trapezoidal rule [43]. The trapezoidal rule is chosen because of the static, uniform grid. The values of the orbitals and the mean fields are only known on the grid points, described by the **R**-matrix in the `grid` -class.

Once an interaction element is calculated, it must be stored. The most obvious way is to use a four dimensional matrix, with $p, q, r$ and $s$ as indices. However, a four-dimensional matrix is an inconvenient data structure. To avoid this problem, a linear mapping scheme, mapping $p, q, r$ and $s$ into one number is introduced. The implementation of the mapping scheme is shown in Listing 8.6. The same mapping scheme is used to store the reduced two-body matrix.

```
#define N1 100
#define N2 10000
#define N3 1000000
int mapTwoParticleStates(int p, int q, int r, int s)
{
    return p + q*N1 + r*N2 + s*N3;
}
```

Listing 8.6: Implementation of the linear scheme mapping the four orbital indices into one number.

To simplify the handling of the interaction elements we store everything that has to with interactions in the `Interaction` -class. The interaction elements are stored in an unordered map, with a unique key generated by the mapping function `mapTwoParticleStates(p,q,r,s)` . If we need an interaction element the function `at(int p, int q, int r, int s)` in the `Interaction` -class is called. It returns the value of the interaction element.

## 8.3 Implementation of the MCTDHF Equations of Motion

In the following subsections we will discuss the implementation of the right hand side of the EOMs, given by Eq. (6.39) and Eq. (6.29). The integration of such equations in time is shown in Chapter 10.

[10] See Chapter 11 for theory and implementation of mean fields.

### 8.3.1 Equation of Motion for the Wavefunction Expansion Coefficients

The equation of motion for the coefficients of the wavefunction is

$$i\frac{\partial \mathbf{A}}{\partial t} = \mathbf{H}\mathbf{A} \ , \tag{8.9}$$

where $\mathbf{A}$ is a coefficient vector and $\mathbf{H}$ is the Hamiltonian set up in the configuration space of the Slater determinants. The matrix elements are given by

$$H_{ij} = \langle \Phi_i | \hat{H} | \Phi_j \rangle \ , \tag{8.10}$$

with $|\Phi_i\rangle$ and $|\Phi_i\rangle$ being Slater determinants. We will now look at how the right hand side of Eq. (8.9) is set up - or more specifically: how the Hamiltonian matrix is constructed using second quantization.

All operations needed to compute the right hand side of Eq. (8.9) are implemented in a class `SlaterEquation`. The `SlaterEquation`-class includes methods for setting up the entire Hamiltonian matrix and calculating the product $\mathbf{HA}$[11]. Note that the `SlaterEquation`-class is general for all Hamiltonian matrices. For testing we have also included functions for diagonalization of the Hamiltonian matrix. Such a test is the same as a CI calculation for that configuration space.

Before discussing the implementation of the Hamiltonian matrix we will look at some optimizations using spin symmetries.

#### Exploiting spin-symmetry

If all the potentials used are spin-independent, we can exploit that two orbitals with identical quantum numbers, except for spin, have the same spatial orbital. We can then pull out the spin degrees of freedom from the Hamiltonian, thus easing some of the computations. Let $p, q, r$ and $s$ denote states with all the quantum numbers excluding spin. Let $\sigma$ and $\gamma$ represent spin quantum numbers. Using such a splitting of quantum numbers yields a one-body operator

$$\hat{h} = \sum_{pq\sigma\sigma'} \langle \phi_{p\sigma} | \hat{h} | \phi_{q\sigma'} \rangle a^\dagger_{p\sigma} a_{q\sigma'} \tag{8.11}$$

$$= \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle \sum_{\sigma\sigma'} a^\dagger_{p\sigma} a_{q\sigma'} \langle \sigma | \sigma' \rangle \tag{8.12}$$

$$= \sum_{pq} \langle \phi_p | \hat{h} | \phi_q \rangle \sum_{\sigma} a^\dagger_{p\sigma} a_{q\sigma} \ . \tag{8.13}$$

[11] To compute the matrix-vector product $\mathbf{HA}$ the C++ library Armadillo [18] is used.

Similarly for the two-body operator:

$$\hat{V} = \frac{1}{2} \sum_{pqrs\sigma\sigma'\delta\delta'} \langle \phi_{p\sigma}\phi_{q\sigma'}|\hat{V}|\phi_{r\delta}\phi_{s\delta'}\rangle a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\delta'}a_{r\delta} \tag{8.14}$$

$$= \frac{1}{2}\sum_{pqrs}\langle \phi_p\phi_q|\hat{V}|\phi_r\phi_s\rangle \sum_{\sigma\sigma'\delta\delta'} a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\delta'}a_{r\delta}\langle\sigma\sigma'|\delta\delta'\rangle \tag{8.15}$$

$$= \frac{1}{2}\sum_{pqrs}\langle \phi_p\phi_q|\hat{V}|\phi_r\phi_s\rangle \sum_{\sigma\sigma'} a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\sigma'}a_{r\sigma} \ . \tag{8.16}$$

The total Hamiltonian can now be written as

$$\hat{H} = \sum_{pq}\langle \phi_p|\hat{h}|\phi_q\rangle \sum_\sigma a^\dagger_{p\sigma}a_{q\sigma} + \frac{1}{2}\sum_{pqrs}\langle \phi_p\phi_q|\hat{V}|\phi_r\phi_s\rangle \sum_{\sigma\sigma'} a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\sigma'}a_{r\sigma} \ . \tag{8.17}$$

All the potentials used in this thesis are spin-independent, and therefore spin symmetry is exploited in the program.

### Matrix Representation of the Hamiltonian

In the equations of motion for the coefficient vector, **A**, the Hamiltonian is represented in the configuration space of the Slater determinants. Using the form of the Hamiltonian given in Eq. (8.17) gives matrix elements

$$H_{ij} = \langle \Phi_i|\hat{H}|\Phi_j\rangle \tag{8.18}$$

$$= \sum_{pq} h_{pq} \sum_\sigma \langle \Phi_i|a^\dagger_{p\sigma}a_{q\sigma}|\Phi_j\rangle + \frac{1}{2}\sum_{pqrs} V_{pqrs} \sum_{\sigma\sigma'}\langle \Phi_i|a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\sigma'}a_{r\sigma}|\Phi_j\rangle \ ,$$

where

$$h_{pq} = \langle \phi_p|\hat{h}|\phi_q\rangle \ , \tag{8.19}$$

$$V_{pqrs} = \langle \phi_p\phi_q|\hat{V}|\phi_r\phi_s\rangle \ . \tag{8.20}$$

We can also exploit that the Hamiltonian matrix is Hermitian, i.e., we only have to compute half the matrix, the other elements are found by taking the Hermitian conjugate. Listing 8.7 shows the construction of the Hamiltonian matrix by solving Eq. (8.18) for every Slater determinant. Notice how the spins sums are hard-coded for electrons in the implementation. Listing 8.8 and Listing 8.9 shows the implementation of $\langle \Phi_i|a^\dagger_{p\sigma}a_{q\sigma}|\Phi_j\rangle$ and of $\langle \Phi_i|a^\dagger_{p\sigma}a^\dagger_{q\sigma'}a_{s\sigma'}a_{r\sigma}|\Phi_j\rangle$.

```
void SlaterEquation::computeHamiltonianMatrix()
{
  cx_double phase;
  cx_double Vpqrs;
  h = &oneParticleOperator->getH();
  H.zeros();

  for(int m=0; m<nSlaterDeterminants; m++){
    for(int n=m; n<nSlaterDeterminants; n++){
      //-----------------------------------------------
      for(int p=0; p<nOrbitals; p++){
        for(int q=0; q<nOrbitals; q++){
          //-----------------------------------------------
          // One-body part

          // The orbitals in the Slater determinants are
              mapped by
          // 2*p == spin up
          // 2*p+1 == spin down
          // where p is the same spatial orbital if p is a
               even number.

          phase = 0;
          phase += secondQuantizationOneBodyOperator(2*p,
              2*q, slaterDeterminants[n],
              slaterDeterminants[m]);
          phase += secondQuantizationOneBodyOperator(2*p
              +1, 2*q+1, slaterDeterminants[n],
              slaterDeterminants[m]);

          H(m,n) += (*h)(p,q)*phase;

          //-----------------------------------------------

          // Two-body interaction
          for(int r=0; r<nOrbitals; r++){
            for(int s=0; s<nOrbitals; s++){
              Vpqrs = interaction->at(p, q, r, s);

              if(Vpqrs != (cx_double) 0 ){
                phase = 0;
                phase += secondQuantizationTwoBodyOperator
                    (2*p, 2*q, 2*r, 2*s, slaterDeterminants
                    [n], slaterDeterminants[m]);

                phase += secondQuantizationTwoBodyOperator
                    (2*p+1, 2*q+1, 2*r+1, 2*s+1,
                    slaterDeterminants[n],
```

```
                    slaterDeterminants[m]);

                phase += secondQuantizationTwoBodyOperator
                    (2*p, 2*q+1, 2*r, 2*s+1,
                    slaterDeterminants[n],
                    slaterDeterminants[m]);

                phase += secondQuantizationTwoBodyOperator
                    (2*p+1, 2*q, 2*r+1, 2*s,
                    slaterDeterminants[n],
                    slaterDeterminants[m]);

                H(m, n) += 0.5*Vpqrs*phase;
              }
              //-----------------------------------------
            }
          }
          //-----------------------------------------
        }
      }
      //-----------------------------------------
      if(m != n){
        H(n, m) = std::conj(H(m,n));
      }
    }
  }
}
```

Listing 8.7: The construction of the Hamiltonian matrix in the Slater determinant space.

```
cx_double SlaterEquation::
    secondQuantizationOneBodyOperator(const int p, const
    int q, bitset<BITS> state1,  const bitset<BITS> &state2
    )
{
  cx_double phase = 1;

  removeParticle(q, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(q, state1);

  addParticle(p, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(p, state1);
```

```
  if (state2 != state1)
    phase = 0;

  return phase;
}
```

Listing 8.8: Implementation of the 'secondQuantizationOneBodyOperator'-function used in the construction of the Hamiltonian matrix. State1 and State2 are Slater determinants.

```
cx_double SlaterEquation::
    secondQuantizationTwoBodyOperator(const int p, const
    int q, bitset<BITS> state1, const bitset<BITS> &state2)
{
  cx_double phase = 1;

  // Removing r
  removeParticle(r, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(r, state1);

  // Removing s
  removeParticle(s, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(s, state1);

  // Adding q
  addParticle(q, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(q, state1);

  // Adding p
  addParticle(p, state1);
  if(state1[BITS-1] == 1)
    return 0;
  phase *= sign(p, state1);

  if (state1 != state2)
    return 0;

  return phase;
}
```

Listing 8.9: Implementation of the 'secondQuantizationTwoBodyOperator'-function used in the construction of the Hamiltonian matrix. State1 and State2 are Slater determinants.

### 8.3.2 The Orbital Equations of Motion

The equations of motion for the orbitals[12] are given by the matrix equation

$$i\frac{\partial \mathbf{C}}{\partial t} = (\mathbb{1} - \mathbf{P})\left[\mathbf{hC} + \boldsymbol{\rho}^{-1}\mathbf{O}\right] \ . \tag{8.21}$$

We will now set up the right hand side of Eq. (8.21). To simplify the implementation we define the matrix

$$\mathbf{U} = \mathbf{hC} + \boldsymbol{\rho}^{-1}\mathbf{O} \ , \tag{8.22}$$

where the columns are given by

$$U_{:,j} = \mathbf{hc}_j + \sum_{iqrs} \rho^{-1}_{ji}\rho_{iqrs}\mathbf{V}^{qr}\mathbf{c}_s \ , \tag{8.23}$$

where $\mathbf{h}$ is the one-body Hamiltonian, $\rho_{ij}$ and $\rho_{iqrs}$ are matrix elements of the reduced one- and two-body density matrices, and $\mathbf{c}_j$ is the spatial discretization of the $j$-th orbital. Before computing the $\mathbf{U}$-matrix, the reduced one- and two-body matrices must be calculated. Details of the implementation of the density operators are in Subsection 8.3.3. After the density matrices have been set up the $\mathbf{U}$-matrix is constructed. The only thing left is the matrix product $(\mathbb{1}-\mathbf{P})\mathbf{U}$. This is explained in the paragraph concerning the projection operator, found further down this section. All operations needed to compute the right hand side of Eq. (8.21) are implemented in the `OrbitalEquation` -class.

#### Constructing the U-matrix

After the construction of the reduced density matrices, the $\mathbf{U}$-matrix is built. The $\mathbf{U}$ matrix is made column by column, as described in Eq. (8.23). The numerical implementation is shown in Listing 8.10.

```
 void OrbitalEquation::computeUMatrix(const cx_mat &C)
{
```

[12] See Chapter 6 for details.

```
    cx_vec Ui(nGrid);
    cx_vec inner(nGrid);
    pair<int,cx_double> rho_iqrs;

    for(int j=0; j<nOrbitals; j++){
      U.col(j) = hC->col(j);
      for(int i=0; i<nOrbitals; i++){
        Ui.zeros();
        for(int q=0; q<nOrbitals; q++){
          for(int r=0; r<nOrbitals; r++){
            inner.zeros();
            for(int s=0; s<nOrbitals; s++){
              inner += findRho2(i,q,r,s) * C.col(s);
            }
            Ui += V->meanField(q,r)%inner;
          }
        }
        U.col(j) += invRho(j,i)*Ui;
      }
    }
  }
```

Listing 8.10: Numerical implementation of the **U**-matrix. The $U$-matrix is a part of the construction of the right hand side of the EOMs for the orbitals. Notice an extensive use of functionality from the matrix library Armadillo.

### The Projection operator

The projection operator, $\hat{P}$, defined as

$$\hat{P} = \sum_{j=1}^{N_{orb}} |\phi_j\rangle \langle\phi_j| \ , \tag{8.24}$$

for the space spanned by the orbitals. We are, however, using orbitals discretized on a grid. The discretized version of the orbitals are stored in the **C**-matrix, where each column, $\mathbf{c}_i$, represents a discretized orbital. Using the discretized grid, the projection operator is a matrix, $\mathbf{P} \in \mathbb{C}^{N_{grid} \times N_{grid}}$, with matrix elements

$$P_{ij} = \left( \sum_k \mathbf{c}_k^T \mathbf{c}_k \right)_{ij} \ . \tag{8.25}$$

In one-dimension this is no problem, but for two- or more dimension the size of **P** in the memory of the computer is very large[13]. When computing larger

[13] An $N_{grid} = 200$ in one dimension amounts to about 625 kB, assuming complex doubles are used. The same resolution in two-dimensions, that is $N_{grid} = 250000$, amounts to a

systems we cannot store this matrix in memory.

What we actually want to solve is the matrix product R.H.S. $= (\mathbb{1}-\mathbf{P})\mathbf{U}$. What if we compute the matrix-matrix product on the fly instead? To do that, we need the residual projection operator, $\mathbf{Q}$, defined as

$$\mathbf{Q} = (\mathbb{1} - \mathbf{P}) \ , \tag{8.26}$$

making the problem R.H.S. $= \mathbf{QU}$. The matrix elements of $\mathbf{Q}$ is given by

$$Q_{ij} = \delta_{ij} - \left( \sum_k \mathbf{c}_k \mathbf{c}_k^T \right)_{ij} = \delta_{ij} - \sum_k^{N_{orb}} C_{ki} C_{kj}^* \ . \tag{8.27}$$

The matrix elements of the matrix-matrix multiplication can be written as

$$(\text{R.H.S.})_{ij} = (QU)_{ij} = \sum_{k=1}^{N_{grid}} Q_{ik} U_{kj}. \tag{8.28}$$

In order to improve performance and reduce memory consumption the product between $\mathbf{Q}$ and $\mathbf{U}$ are hard-coded into the function `computeRightHandSide()` by using Eqs. (8.28) and (8.27). Listing 8.11 shows how this is implemented in the code. We are now using approximately no memory, we don't have to build the $\mathbf{Q}$-matrix and then perform the multiplication, making this implementation faster since $\mathbf{Q}$ is not used for anything else in the code.

```
const cx_mat &OrbitalEquation::computeRightHandSide(const
    cx_mat &C, const cx_vec &A){
...
  //-----------------------------------------------------------
  // Hardcoding of the matrix-matrix product: R.H = Q*U
  //-----------------------------------------------------------
  cx_double RH_ij;
  cx_double Qik;
  const cx_double *C_ = C.memptr();
  cx_double *U_ = U.memptr();
  cx_double *RH_ = rightHandSide.memptr();

  int i, j;
  for(pair<int,int> ij :myRij){
      i = ij.first;
      j = ij.second;

      RH_ij = 0;
      for(int k=0; k<nGrid; k++){
```

staggering 23.8 GB.

```
            // Calculating the Residual projector
            Qik = 0;
            if(i == k)
                Qik = 1;
            for(int l=0; l<nOrbitals; l++){
                Qik -= C_[i + l*nGrid]*conj(C_[k + l*nGrid])
                    ;
            }
            // END projector
            RH_ij += Qik*U_[k + j*nGrid];
        }
        RH_[i + j*nGrid] = RH_ij;
...
}
```

Listing 8.11: Implementation of the matrix product **QU** used to compute the right hand side of the EOM for the orbitals.

### 8.3.3 The Reduced Density Operators

In the equation of motion for the orbitals the reduced one- and two-body density matrices enter. In the following subsection we will discuss their implementation.

#### Reduced Density Matrices

The reduced one-body density is[14]

$$\rho(\mathbf{r}_1|\mathbf{r}_1') = \sum_{pq} \rho_{pq}\phi_p^\dagger(\mathbf{r}_1)\phi_q(\mathbf{r}_1') \ , \tag{8.29}$$

with the reduced one-body density matrix elements

$$\rho_{ij} = \langle\Psi|a_i^\dagger a_j|\Psi\rangle = \sum_{nm} A_n^* A_m \langle\Phi_n|a_i^\dagger a_j|\Phi_m\rangle \ . \tag{8.30}$$

The reduced two-body density is

$$\rho(\mathbf{r}_1,\mathbf{r}_2|\mathbf{r}_1',\mathbf{r}_2') = \sum_{pqrs} \rho_{pqrs}\phi_p^\dagger(\mathbf{r}_1)\phi_q^\dagger(\mathbf{r}_2)\phi_r(\mathbf{r}_2')\phi_s(\mathbf{r}_1') \ , \tag{8.31}$$

with the reduced two-body density matrix elements

$$\rho_{pqsr} = \langle\Psi|a_p^\dagger a_q^\dagger a_s a_r|\Psi\rangle = \sum_{nm} A_n^* A_m \langle\Phi_n|a_p^\dagger a_q^\dagger a_s a_r|\Phi_m\rangle \ . \tag{8.32}$$

[14] See Section 3.4

### Spin Reduced Density Matrices

We will now perform some changes to the optimization of the equations of motion for spin independent Hamiltonians. As a part of the derivation of the MCTDHF equations[15] the expression $\langle\Psi|\hat{H}|\Psi\rangle$ is written as a function of the reduced density matrices. We will now modify the equations of motion slightly by separating out the spin degrees of freedom. First, we look at the expectation value of the one-body Hamiltonian. Let $p, q, r$ and $s$ denote states with all the quantum numbers excluding spin. Let $\sigma$ and $\gamma$ represent spin quantum numbers. With such as splitting of the quantum numbers we get

$$\langle\Psi|\hat{h}|\Psi\rangle = \sum_{pq\sigma\sigma'} \rho_{p\sigma q\sigma'}\langle\phi_{p\sigma}|\hat{h}|\phi_{q\sigma'}\rangle = \sum_{pq}\langle\phi_p|\hat{h}|\phi_q\rangle \sum_{\sigma\sigma'} \rho_{p\sigma q\sigma'}\langle\sigma|\sigma'\rangle \ . \quad (8.33)$$

The last element is the reduced one-body density matrix with the trace taken over the spins. We define it as

$$\rho_{pq} = \sum_{\sigma\sigma'} \rho_{p\sigma q\sigma'}\langle\sigma|\sigma'\rangle = \sum_{\sigma} \rho_{p\sigma q\sigma} \ . \quad (8.34)$$

For the two-body operator:

$$\langle\Psi|\hat{V}|\Psi\rangle = \frac{1}{2}\sum_{pqrs\sigma\sigma\delta\delta'} \rho_{p\sigma q\sigma' r\delta s\delta'}\langle\phi_{p\sigma}\phi_{q\sigma'}|\hat{V}|\phi_{s\delta'}\phi_{r\delta}\rangle \quad (8.35)$$

$$= \frac{1}{2}\sum_{pqrs}\langle\phi_p\phi_q|\hat{V}|\phi_s\phi_r\rangle \sum_{\sigma\sigma'\delta\delta'} \rho_{p\sigma q\sigma' r\delta s\delta'}\langle\sigma\sigma'|\delta'\delta\rangle \ . \quad (8.36)$$

Likewise, the last element is the reduced density matrix with the trace taken over the spins. We define it as

$$\rho_{pqrs} = \sum_{\sigma\sigma'\delta\delta'} \rho_{p\sigma q\sigma' r\delta s\delta'}\langle\sigma\sigma'|\delta'\delta\rangle = \sum_{\sigma\sigma'} \rho_{p\sigma q\sigma' r\sigma' s\sigma} \ . \quad (8.37)$$

We have shown that in the equations of motion for a spin-independent Hamiltonians we can use the spin reduced density matrices instead of the full density matrices.

### Implementation of the Reduced Density Matrices

We can check the implementation of the reduced density matrices by taking the trace. For the reduced one-body density matrix the trace is $\mathrm{Tr}(\rho^{(1)}) = N_p$.

[15] See Chapter 6.

For the reduced two-body density matrix the trace is $\mathrm{Tr}(\rho^{(2)}) = N_p(N_p - 1)$. Listing 8.12 and Listing 8.13 shows the implementation of the reduced density matrices in the code.

```
 void OrbitalEquation::computeOneParticleReducedDensity()
{
  // All possible spatial orbitals
  for(int i=0; i< nOrbitals; i++){
    for(int j=i; j<nOrbitals; j++){
      invRho(i,j)  = reducedOneParticleOperator(2*i,2*j);
      invRho(i,j)  += reducedOneParticleOperator(2*i+1,2*j
          +1);
      invRho(j,i) = conj(invRho(i,j));
    }
  }
}
```

Listing 8.12: Implementation of the one-body reduced density matrix.

```
void OrbitalEquation::computeTwoParticleReducedDensity()
{
  cx_double value;

  // All different
  for(int p=0; p<nOrbitals; p++){
    for(int q=0; q<nOrbitals; q++){
      for(int r=0; r<nOrbitals; r++){
        for(int s=0; s<nOrbitals; s++){
          value = 0;

          // Spin trace
          value += reducedTwoParticleOperator(2*p, 2*q, 2*
              r, 2*s);
          value += reducedTwoParticleOperator(2*p+1, 2*q
              +1, 2*r+1, 2*s+1);

          value += reducedTwoParticleOperator(2*p+1, 2*q,
              2*r, 2*s+1);
          value += reducedTwoParticleOperator(2*p, 2*q+1,
              2*r+1, 2*s);

          rho2.insert( pair<int,cx_double>(
              mapTwoParticleStates(p,q,r,s), value) );
        }
      }
    }
  }
}
```

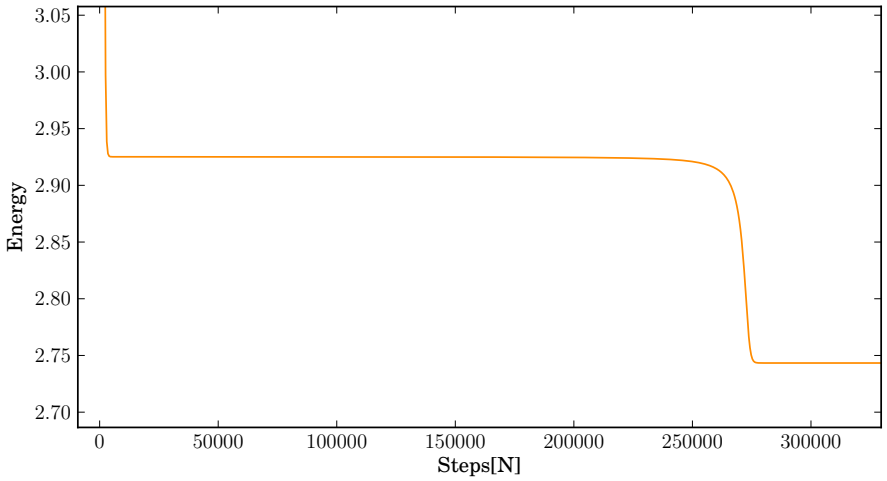



Figure 8.3: An example of an apparent convergence of the energy.

Listing 8.13: Implementation of the two-body reduced density matrix.

## 8.4 Convergence Criteria

To find an initial state we use the imaginary time propagation method, giving us the ground state of the time-independent problem. As the imaginary time approach is an iterative method, we need to measure whether a calculation has converged. When performing imaginary time calculations, we compute the energy of the system at runtime. After performing a number of integration steps, the energy will start to converging towards a constant value. As a test of convergence, a newly calculated value for the energy is compared with the energy calculated 100 steps before. We call this difference $\Delta E$. If $\Delta E < 10^{-14}$, we say that the system has converge to numerical precession. The reason $\Delta E$ is chosen so low, is that we sometimes get an apparent convergence of the energy, like seen in Figure 8.3. The $\Delta E$ can be as low as $10^{-9}$, and the system might still not have converged. This usually happen if the initial wavefunction only has a small overlap with the ground state, and/or if the difference in energy between the ground state and the first excited state is very small. Whenever a result is marked with a star, $*$, the results have not met the strict convergence criteria of $\Delta E < 10^{-14}$.

| Cores | Time[s] |
|---|---|
| 1 | 319 |
| 4 | 123 |
| 20 | 42 |
| 30 | 46 |

(a) Time spent calculating a sequence of imaginary time steps using different number of compute cores. The system example has six orbitals and a grid consisting of 10000 elements.

| Cores | Time[s] |
|---|---|
| 20 | 379 |
| 36 | 175 |
| 40 | 164 |
| 64 | 233 |

(b) Time spent calculating a sequence of imaginary time steps using different number of compute cores. The example system has six orbitals and a grid consisting of 22500 elements.

Table 8.1: Some examples of efficiency of the parallelized code.

## 8.5 Parallelization

The code is parallelized using OpenMPI [44]. For the computationally demanding parts of the code, the workload is spread over a multitude of computer nodes. Typical examples of the numerically demanding parts is the computation of mean fields, matrix elements, matrix multiplication, etc. At the start of the program, every node is given a set of orbital- and spatial-indices for which it is to compute, for example, matrix elements or mean fields. These are stored in STD[16] vectors, making the implementation general and efficient. The set of indices may vary in the different classes. Every node runs the full program parallel, and at every time step the full orbital matrix, $\mathbf{C}$, all mean fields, and interaction elements are communicated to every node. This is due to the coupled nature of the computations, and it puts an effective barrier on the number of nodes the program can efficiently use. The mean fields take between 40-60 % of the computation time, and are the most important part to parallelize. The parallelization implemented in the code is a temporary solution, however, the computation of two-dimensional systems would not have been possible without this parallelization. For a more complete scheme for the parallelization of MCTDHF method, see reference [45]. Some example of the efficiency of the parallelization is shown in Table 8.1. The results are very system dependent, and the code scales better for larger systems.

[16] The Standard Template Library, or STL, is a C++ library of container classes, algorithms, and iterators [40].

# Chapter 9

# Representation of Differential Operators

In this chapter we discuss the implementation of the differential operators used to calculate the second derivative of the orbitals. We present two different methods used in the approximation of the differential operators: finite difference methods and Fourier transformations. The theory and implementation of both methods are discussed and example implementations are given.

## 9.1 Representation of Differential Operators

The implementation of differential operators is very similar to that of the potentials. The differential operators have been separated from the potentials due to the structure the code, but could just as easily have been implemented in the `potential` class. The general form of the implementation is given by the abstract base class, `DifferentialOperator`. Explicit implementations of differential operators are subclassed from the `DifferentialOperator` class, and they must include a virtual function called `secondDerivative(const cx_vec &phi)`, where `phi` is a reference to a spatial discretized orbital. This function must return a vector containing the second derivative of `phi`.

## 9.2 Finite Difference

Finite difference schemes are based on Taylor expansions of different orders. For example, a first order Taylor expansion of a function $f(x)$ about $x_i$ is

$$f(x) = f(x_i) + f'(x_i)(x - x_i) + \mathcal{O}((x - x_i)^2) \ . \tag{9.1}$$

Solving for the first derivative gives us

$$f'(x_i) = \frac{f(x) - f(x_i)}{x - x_i} + \mathcal{O}(x - x_i) \ . \tag{9.2}$$

If we are using an uniform grid spacing, we can use that $x = x_i + \Delta x = x_{i+1}$. We will also use the notation $f_i = f(x_i)$. This gives us

$$f'_i = \frac{f_{i+1} - f_i}{\Delta x} + \mathcal{O}(\Delta x) \ . \tag{9.3}$$

This scheme is called a *forward difference approximation* or simply forward Euler and is the simplest scheme for calculating a derivative numerically. Likewise we can find the *backwards difference approximation*

$$f'_i = \frac{f_i - f_{i-1}}{\Delta x} + \mathcal{O}(\Delta x) \ . \tag{9.4}$$

Adding the two schemes together and we get the *central difference approximation*:

$$f'_i = \frac{f_{i+1} - f_{i-1}}{2\Delta x} + \mathcal{O}(\Delta x^2) \ . \tag{9.5}$$

We get an error $\mathcal{O}(\Delta x^2)$ due to a cancellation of the leading order terms. Expanding the error to one higher order in the forward and backward schemes shows that this is true.

We are actually just interested in the second derivative. The forward finite difference scheme for the second derivative is found by performing a change of variables $f' \to f''$ in Eq. (9.3), and results in

$$f''_i = \frac{f_{i+2} - 2f_{i-1} + f_i}{\Delta x^2} + \mathcal{O}(\Delta x) \ . \tag{9.6}$$

The central difference approximation is

$$f''_i = \frac{f_{i+1} - 2f_i + f_{i-1}}{\Delta x^2} + \mathcal{O}(\Delta x^2) \ . \tag{9.7}$$

Another name for this approximation is the *three point stencil*. In the code we have implemented a three point stencil and a five-point stencil.

### Three-Point Stencil

Using the three-point stencil we can approximate the second derivative of an orbital in the $x$-direction by

$$\nabla_x^2 \phi(x_i) \approx \frac{\phi(x_{i+1}) - 2\phi(x_i) + \phi(x_{i-1})}{\Delta x^2} + \mathcal{O}(\Delta x^2) \ , \tag{9.8}$$

where $x_{i\pm1} = x_i \pm \Delta x$ and $\Delta x$ is the grid spacing.

Listing 9.1 shows the numerical implementation of a three-point stencil.

```
 cx_vec FiniteDifference2d::secondDerivative(const cx_vec
    &phi)
{
    diff.zeros();

    cx_double diffX;
    cx_double diffY;

    for(int i=0; i<nGrid; i++){

        // Center point
        diffX = -(cx_double)2*phi(i);
        diffY = diffX;

        // X ------------------------
        if(i + nGridX < nGrid)
            diffX += phi(i + nGridX);

        if(i - nGridX > 0)
            diffX += phi(i - nGridX);

        // Y ------------------------
        if(i + 1 < nGrid)
            diffY += phi(i + 1);

        if(i - 1 > 0)
            diffY += phi(i - 1);

        diff(i) = diffX/dxdx + diffY/dydy;
    }

    return diff;
}
```

Listing 9.1: Implementation of the second derivative of an orbital using a three-point stencil in two-dimensions.

### Five-Point Stencil

Using the five-point stencil we can approximate the second derivative of an orbital in the $x$-direction by

$$\frac{\partial^2\phi(x_i)}{\partial x_i^2} \approx \frac{-\phi(x_{i+2}) + 16\phi(x_{i+1}) - 30\phi(x_i) + 16\phi(x_{i-1}) - \phi(x_{i-2})}{12\Delta x^2} + \mathcal{O}(\Delta x^4) \ . \tag{9.9}$$

The implementation is basically the same as the three-point stencil, only with more terms.

## 9.3 Discrete Fourier transforms (DFT)

Using the Discrete Fourier Transformation (DFT) theory a function, $\phi$, evaluated in a set of discrete points $\{x_l\}$, can be expressed in the frequency domain by

$$\phi(x_l) = \phi_l = \sum_{k=0}^{N-1} \exp\left(2\pi i \frac{kl}{N}\right) \hat{f}(k) \ , \tag{9.10}$$

where the $\hat{f}(k)$ is the discrete Fourier transform of $\phi_l$ and $N$ is the number of grid points in one-dimension. The Fourier transform is defined as

$$\hat{f}(k) = \frac{1}{N} \sum_{l=0}^{N-1} \exp\left(-2\pi i \frac{kl}{N}\right) \phi_l \ . \tag{9.11}$$

We can exploit the Fourier transform to calculate the derivatives of $\phi$ with a higher precision than finite difference methods. Taking the second derivative is $\phi_l$:

$$\nabla^2_{x_l} \phi_l = \sum_{k=0}^{N-1} \nabla^2_{x_l} \exp\left(2\pi i \frac{kl}{N}\right) \hat{f}(k) \ . \tag{9.12}$$

We use that

$$\frac{d}{dx_l} \exp\left(2\pi i \frac{kl}{N}\right) = \frac{dl}{dx_l} \frac{d}{dl} \exp\left(2\pi i \frac{kl}{N}\right) = i \frac{2\pi k}{N} \frac{dl}{dx_l} \exp\left(2\pi i \frac{kl}{N}\right) \ , \tag{9.13}$$

where

$$x_l = x_0 + \Delta x l, \quad \Rightarrow \frac{dl}{dx_l} = \frac{1}{\Delta x} \ , \tag{9.14}$$

for an uniform grid. The second derivative can now be expressed as

$$\nabla^2_{x_l} \phi_l = -\left(\frac{2\pi}{N\Delta x}\right)^2 \sum_{k=0}^{N-1} \exp\left(2\pi i \frac{l}{N}\right) k^2 \hat{f}(k) \ . \tag{9.15}$$

Equation (9.15) shows us that taking the inverse transform of $-\left(\frac{2\pi}{N\Delta x}\right)^2 k^2 \hat{f}(k)$ gives us the second derivative - just as we wanted. To summarise: first compute the Fourier transform $\hat{f}(k)$. Then take the inverse Fourier transform of $-\left(\frac{2\pi}{N\Delta x}\right)^2 k^2 \hat{f}(k)$, and we have the second derivative.

When using Fourier transforms the boundary conditions must be periodic - which might affect the physical system if the wavefunction approaches one of the borders on the grid. The error introduced by using discrete Fourier transformations is of the order $\mathcal{O}(\Delta x^{N_{grid,d}})$ [46], where $N_{grid,d}$ is the number of grid points in one dimension. To efficiently compute discrete Fourier transformations the Fast Fourier Transform (FFT) algorithm[1] is used. In the code developed in this thesis the FFT-library FFTW [47] is used.

Using Fourier transformations to compute the derivative does, however, come at a cost. The geometry of the grid must be equally spaced, making the method less appropriate for more systems using a more complex spatial geometry.

Listings 9.2 shows the implementation of the second derivative of a two-dimensional orbital, using the FFTW library integrated with Armadillo.

```
 Fourier2d::Fourier2d(Config* cfg, const Grid &grid):
    DifferentialOperator(cfg, grid)
{
  ...
  // Loading paramteres from the configuration object
  ...
  // Setting the frequencies.
  // X
  vec k_x = vec(nGridX);

  for(int i=0; i<nGridX/2; i++) {
    k_x[i] = i;
  }
  for(int i=nGridX/2; i<nGridX;i++) {
    k_x[i] = -(nGridX - i);
  }

  k_x *= 2*PI/(dx*(nGridX));
  k_x = -k_x%k_x;

  // Y
  vec k_y = vec(nGridY);

  for(int i=0; i<nGridY/2; i++) {
    k_y[i] = i;
  }
  for(int i=nGridY/2; i<nGridY;i++) {
    k_y[i] = -(nGridY - i);
  }

  k_y *= 2*PI/(dy*(nGridY));
```

[1] The FFT algorithm requires $\mathcal{O}(N \log N)$ operations.

```
  k_y = -k_y%k_y;

  // Creating the total frequency space
  k = vec(nGrid);
  int c = 0;
  for(int i=0; i<nGridX; i++) {
    for(int j=0; j<nGridY; j++) {
      k(c++) = k_x(i) + k_y(j);
    }
  }
  k /= nGrid;

  diff = cx_vec(nGrid);

  forward = fftw_plan_dft_2d(nGridX, nGridY,
    (fftw_complex*)diff.memptr(),
    (fftw_complex*)diff.memptr(), FFTW_FORWARD,
        FFTW_ESTIMATE);
  backward = fftw_plan_dft_2d(nGridX, nGridY,
    (fftw_complex*)diff.memptr(),
    (fftw_complex*)diff.memptr(), FFTW_BACKWARD,
        FFTW_ESTIMATE);
}

cx_vec Fourier2d::secondDerivative(const cx_vec &phi)
{
  diff = phi;
  fftw_execute(forward);
  diff = k % diff;
  fftw_execute(backward);
  return diff;
}
```

Listing 9.2: Implementation of a two-dimensional Fourier transform for computation of the second derivative of an orbital.

# Chapter 10

# Integration Schemes

In this chapter numerical schemes for integrating the MCTDHF equations of motion are presented. The equations of motion are given through a set of non-linear, partial, coupled differential equations, and a robust integrator is needed to solve them. We start with an illustrative example of the explicit Euler integration scheme to describe the basic ideas and implementation. Afterwards we give a review of the Runge-Kutta integrators, a family of finite difference integrators. The integrators we ended up using are the fourth order Runge-Kutta and the adaptive Runge-Kutta-Fehlberg method. A short discussion of other integrators is found in the last section.

## 10.1 Time Propagation

An abstract class `TimePropagation` is used to handle time propagation. The time propagation starts when the function `doTimePropagation` in `TimePropagation` is called. The basic code flow is shown in Listing 10.1. The function `stepForward` is called at every iteration, and contains the explicit implementation of an integrator algorithm. To implement an integration scheme a subclass of `TimePropagation` is created, and the function `stepForward` must be implemented, containing the algorithm of choice.

```
void TimePropagation::doTimePropagation()
{
  int counter = 0;
  bool accepted;

  for(step=0; step < N; step++){
    accepted = this->stepForward();

    // Saving C and A to disk
```

```
    if(accepted){
      if((step % saveToFileInterval == 0 || step == N-1) )
         {
          time(counter) = t;

#ifdef USE_MPI
          MPI_Bcast( C.memptr(), C.n_elem ,
              MPI_DOUBLE_COMPLEX, 0, MPI_COMM_WORLD );
#endif

          // Updating the one- and two-body interaction-
              elements
          V->computeNewElements(C);
          h->computeNewElements(C, t);

          // Collecting data
          if(isMaster){
            E(counter) = slater->getEnergy(A);
            rho = &orbital->reCalculateRho1(A);
            K = orbital->getCorrelation();
            svdRho = orbital->getSvdRho1();

            saveProgress(counter);
            printProgressToScreen(counter);
          }
          counter++;
      }
      t += dt;
    }
  }
}
// Saving results
if(isMaster)
  time.save(filenameT, arma_ascii);
```

Listing 10.1: Implementation of the time propagation loop.

## 10.2 Integration Schemes

In the following subsections different integration algorithms are presented. The notation $f_i = f(t_i)$ is used as a shorthand for the $i$-th time step.

### 10.2.1 Explicit Euler

The explicit Euler integration method is *the* simplest scheme one can use for integration, and is shown only as an introduction to the other methods. It is

too unstable and inaccurate to be used for actual computations.

The forward Euler integration scheme is basically a first order Taylor expansion, that is

$$f_{n+1} = f_n + \frac{df_n}{dt}\Delta t + \mathcal{O}(\Delta t^2) \ , \tag{10.1}$$

where $\Delta t$ is a constant increment in time.

As an example, we will apply Euler's methods to the wavefunction's coefficients vector, $\mathbf{A}$, and the orbital coefficient matrix, $\mathbf{C}$:

$$\mathbf{A}_{n+1} = \mathbf{A}_n + \frac{d\mathbf{A}_n}{dt}\Delta t \tag{10.2}$$

and

$$\mathbf{C}_{n+1} = \mathbf{C}_n + \frac{d\mathbf{C}_n}{dt}\Delta t \ . \tag{10.3}$$

The time-derivatives are given by the MCTDHF equations of motion, i.e., Eq (6.39) and Eq. (6.35). Using the MCTDHF equations of motion the next steps are found by:

$$\mathbf{A}_{n+1} = \mathbf{A}_n - i\Delta t\mathbf{H}_n\mathbf{A}_n \tag{10.4}$$

and

$$\mathbf{C}_{n+1} = \mathbf{C}_n - i\Delta t(\mathbb{1} - \mathbf{P}_n)\left[\mathbf{h}_n\mathbf{C}_n + \boldsymbol{\rho}_n^{-1}\mathbf{O}_n\right] \ , \tag{10.5}$$

for the orbitals. This is what we have to implement in the `stepForward` function. We have assumed that the time $\Delta t$ is so short that the time-dependent operators can be assumed to be constant over the interval $\Delta t$.

## 10.3 Runge-Kutta methods

Runge-Kutta methods use the weighted average of the derivative at different points during one time step. As Runge-Kutta methods do not necessarily conserve orthogonality or norm, we have to explicitly enforce these properties in other ways. To enforce orthogonality and conservation of norm in a set of orbitals, a Singular Value Decomposition (SVD) [48] is performed after each accepted step. A renormalization of the coefficient vector, $\mathbf{A}$, is also performed.

### Second Order Runge-Kutta

The second order Runge Kutta (RK2) is based on a second order Taylor expansion:

$$y_{n+1} = y_n + \Delta t\frac{dy_n}{dt} + \frac{\Delta t^2}{2}\frac{d^2y_n}{dt^2} + \mathcal{O}(\Delta t^3) \ . \tag{10.6}$$

The first derivative of $y$ is written as

$$\frac{dy_n}{dt} = f(t, y_n) \ . \tag{10.7}$$

The second derivative can now be expressed as

$$\frac{d^2y_n}{dt^2} = \frac{\partial f(t, y_n)}{\partial t} + \frac{\partial f(t, y_n)}{\partial y_n}\frac{\partial y_n}{\partial t} \ . \tag{10.8}$$

Inserting Eq. (10.7) and Eq. (10.8) into Eq. (10.6), we get

$$y_{n+1} = y_n + \frac{\Delta t}{2} f(t, y_n) + \frac{\Delta t}{2}\left( f(t, y_n) + \Delta t \frac{\partial f(t, y_n)}{\partial t} + \Delta t \frac{\partial f(t, y_n)}{\partial y_n}\frac{\partial y_n}{\partial t}\right) + \mathcal{O}(\Delta t^3) \ . \tag{10.9}$$

By recognizing that the last term is a multi-variable Taylor expansion

$$f(t + \Delta t, y_n + \Delta t f_{t,y_n}) = f(t, y_n) + \Delta t \frac{\partial f_{t,y_n}}{\partial t} + \Delta t \frac{\partial f(t, y_n)}{\partial y_n}\frac{\partial y_n}{\partial t} \ , \tag{10.10}$$

the step can be compressed into

$$y_{n+1} = y_n + \frac{\Delta t}{2}\left(f(t, y_n) + f(t + \Delta t, y_n + \Delta t f_{t,y_n})\right) + \mathcal{O}(\Delta t^3). \tag{10.11}$$

To simplify the equation we define the weights

$$k_1 = \Delta t f(t, y_n) \tag{10.12}$$

$$k_2 = \Delta t f(t + \Delta t, y_n + \Delta t k_1) \ , \tag{10.13}$$

resulting in

$$y_{n+1} = y_n + \frac{1}{2}\left(k_1 + k_2\right) + \mathcal{O}(\Delta t^3). \tag{10.14}$$

which is the usual way of writing the second order Runge-Kutta method. We have found the local error of RK2 to be $\mathcal{O}(\Delta t^3)$, resulting in a second order global error. To calculate a step using RK2, first compute $k_1$, then $k_2$, and insert into Eq. (10.14).

We have not implemented the RK2 method in the program due to a lack in both precision and stability of the method. The derivation was performed to show the basic concept of Runge-Kutta methods.

### Fourth Order Runge-Kutta

The Fourth order Runge-Kutta (RK4) is considered by many to be the standard workhorse integrator. The method is frequently used due to its ease of implementation, but at the same time being numerical accurate. There are several ways to derive the method. Personally, I like to begin with a Taylor expansion

$$y_{n+1} = y_n + \Delta t\frac{dy_n}{dt} + \frac{\Delta t^2}{2}\frac{d^2y_n}{dt^2} + \frac{\Delta t^3}{6}\frac{d^3y_n}{dt^3} + \frac{\Delta t^4}{24}\frac{d^2y_n}{dt^4} + \mathcal{O}(\Delta t^5)\ . \quad (10.15)$$

Following the same line of arguments used for the RK2, we find

$$y_{n+1} = y_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)\ , \quad (10.16)$$

with the weights being

$$k_1 = \Delta t f(t_n, y_n)\ , \quad (10.17)$$
$$k_2 = \Delta t f(t_n + \tfrac{\Delta t}{2}, y_n + \tfrac{\Delta t}{2}k_1)\ , \quad (10.18)$$
$$k_3 = \Delta t f(t_n + \tfrac{\Delta t}{2}, y_n + \tfrac{\Delta t}{2}k_2)\ , \quad (10.19)$$
$$k_4 = \Delta t f(t_n + \Delta t, y_n + k_3)\ . \quad (10.20)$$

The global error is of order $\mathcal{O}(\Delta t^4)$.

To implement RK4 in the program the weights must be expressed for every Slater determinant vector and the single-particle orbitals. For the Slater determinants the next step is

$$\mathbf{A}_{n+1} = \mathbf{A}_n + \frac{1}{6}(\mathbf{k}_1 + 2\mathbf{k}_2 + 2\mathbf{k}_3 + \mathbf{k}_4)\ , \quad (10.21)$$

with the weights being the derivatives, i.e. the

$$\mathbf{k}_1 = -i\mathbf{H}(t_n)\mathbf{A}_n\ , \quad (10.22)$$
$$\mathbf{k}_2 = -i\mathbf{H}(t_n + \tfrac{\Delta t}{2})\left(\mathbf{A}_n + \tfrac{\Delta t}{2}\mathbf{k}_1\right)\ , \quad (10.23)$$
$$\mathbf{k}_3 = -i\mathbf{H}(t_n + \tfrac{\Delta t}{2})\left(\mathbf{A}_n + \tfrac{\Delta t}{2}\mathbf{k}_2\right)\ , \quad (10.24)$$
$$\mathbf{k}_4 = -i\mathbf{H}(t_n + \Delta t)\left(\mathbf{A}_n + \mathbf{k}_3\right)\ . \quad (10.25)$$

Likewise, for the $\mathbf{C}$-matrix containing all the orbitals we get

$$\mathbf{C}_{n+1} = \mathbf{C}_n + \frac{1}{6}(\mathbf{m}_1 + 2\mathbf{m}_2 + 2\mathbf{m}_3 + \mathbf{m}_4)\ , \quad (10.26)$$

and the weights are

$$\mathbf{m}_1 = \Delta t R_c(t_n, \mathbf{C}_n) \ , \tag{10.27}$$
$$\mathbf{m}_2 = \Delta t R_c(t_n + \tfrac{\Delta t}{2}, \mathbf{C}_n + \tfrac{\Delta t}{2}\mathbf{m}_1) \ , \tag{10.28}$$
$$\mathbf{m}_3 = \Delta t R_c(t_n + \tfrac{\Delta t}{2}, \mathbf{C}_n + \tfrac{\Delta t}{2}\mathbf{m}_2) \ , \tag{10.29}$$
$$\mathbf{m}_4 = \Delta t R_c(t_n + \Delta t, \mathbf{C}_n + \mathbf{m}_3) \ . \tag{10.30}$$

and $R_c$ is the right hand side of Eq. (6.35):

$$R_c(t, \mathbf{C}) = -i(\mathbb{1} - \mathbf{P}(\mathbf{C}))\left[\mathbf{h}(t)\mathbf{C} + \boldsymbol{\rho}^{-1}\mathbf{U}(\mathbf{C})\right] \ , \tag{10.31}$$

Listing 10.2 shows the implementation of RK4 in the program.

```
bool RungeKutta4::stepForward()
{
  // Computing the Runge-Kutta weights
  // Notice that the one- and two-body operator must be
     updated for each weight

  V->computeNewElements(C);
  h->computeNewElements(C, t);

  k1 = -i*dt*slater->computeRightHandSide(A);
  m1 = -i*dt*orbital->computeRightHandSide(C, A);

  V->computeNewElements(C + 0.5*m1);
  h->computeNewElements(C + 0.5*m1, t + 0.5*dt);

  k2 = -i*dt*slater->computeRightHandSide(A + 0.5*k1);
  m2 = -i*dt*orbital->computeRightHandSide(C + 0.5*m1, A +
       0.5*k1);

  V->computeNewElements(C + 0.5*m2);
  h->computeNewElements(C + 0.5*m2, t + 0.5*dt);

  k3 = -i*dt*slater->computeRightHandSide(A + 0.5*k2);
  m3 = -i*dt*orbital->computeRightHandSide(C + 0.5*m2, A +
       0.5*k2);

  V->computeNewElements(C + m3);
  h->computeNewElements(C + m3, t + dt);

  k4 = -i*dt*slater->computeRightHandSide(A + k3);
  m4 = -i*dt*orbital->computeRightHandSide(C + m3, A + k3)
      ;

  // Computing new states
```

```
    A += 1.0/6.0*(k1 + 2*(k2 + k3) + k4);
    C += 1.0/6.0*(m1 + 2*(m2 + m3) + m4);

    // Re-normalizing
    renormalize(C);
    A = A/sqrt(cdot(A,A));

    return 1; // The step is always accepted
}
```

Listing 10.2: Implementation of the fourth order Runge-Kutta integrator.

### 10.3.1 Runge-Kutta-Fehlberg

Runge-Kutta-Fehlberg uses a Runge-Kutta solver of degree $p$ and $p+1$ to estimate the local truncation error during propagation by comparing the results. From the error estimate the step length is chosen so the error is always below an error threshold $\varepsilon$, making the method *adaptive* during runtime. For our calculations we will use a degree $p = 4$, meaning that we will compute a RK4 and compare with a RK5. The shorthand notation for this implementation is RK45. To efficiently implement to RK45 it is important to choose the weights so that the RK4 and RK5 shares the same weights. Otherwise the scheme would be computationally inefficient.

The fourth order approximation is given by

$$y'_{i+1} = y_i + \frac{25}{216}k_1 + \frac{1408}{2565}k_3 + \frac{2197}{4104}k_4 - \frac{1}{5}k_5 \ , \tag{10.32}$$

and $\tilde{y}_{i+1}$ is the fifth order approximation

$$\tilde{y}_{i+1} = y_i + \frac{16}{135}k_1 + \frac{6656}{12825}k_3 + \frac{28561}{56430}k_4 - \frac{1}{5}k_5 + \frac{2}{55}k_6 \ . \tag{10.33}$$

The Runge-Kutta weights are

$$k_1 = \Delta t f(t_i, y_i) \ , \tag{10.34}$$

$$k_2 = \Delta t f\left(t_i + \frac{\Delta t}{4}, y_i + \frac{1}{4}k_1\right) \ , \tag{10.35}$$

$$k_3 = \Delta t f\left(t_i + \frac{3\Delta t}{8}, y_i + \frac{3}{32}k_1 + \frac{9}{32}k_2\right) \ , \tag{10.36}$$

$$k_4 = \Delta t f\left(t_i + \frac{12\Delta t}{13}, y_i + \frac{1932}{2197}k_1 - \frac{7200}{2197}k_2 + \frac{7296}{2197}k_3\right) \ , \tag{10.37}$$

$$k_5 = \Delta t f\left(t_i + \Delta t, y_i + \frac{439}{216}k_1 - 8k_2 + \frac{3680}{513}k_3 - \frac{845}{4104}k_4\right) \ , \tag{10.38}$$

$$k_6 = \Delta t f\left(t_i + \frac{\Delta t}{2}, y_i - \frac{8}{27}k_1 + 2k_2 - \frac{3544}{2565}k_3 + \frac{1859}{4104}k_4 - \frac{11}{40}k_5\right) \ . \tag{10.39}$$

The results of the RK4 and RK5 is used to estimate the error

$$R = \frac{1}{\Delta t}|\tilde{y}_{i+1} - y'_{i+1}| \ , \tag{10.40}$$

and a time-scaling factor $\delta$ is found by

$$\delta = 0.84\left(\frac{\varepsilon}{R}\right)^{1/4} \ . \tag{10.41}$$

After a new step taken, the steps are accepted or rejected according to

- if $R \leq \varepsilon$ : set $y_{n+1} = y'_{i+1}$ and let the step size $\Delta t \to \delta \Delta t$ .
- if $R > \varepsilon$ : recalculate the current step with step size $\Delta t \to \delta \Delta t$ .

The equations of motion are set up the same way as we did for RK4. Listing 10.3 shows how RK45 is implemented for the MCTDHF method.

```
bool RungeKuttaFehlberg::stepForward()
{
  cx_vec k1, k2, k3, k4, k5, k6;
  cx_mat m1, m2, m3, m4, m5, m6;
  cx_vec A_, A_1;
  cx_mat C_, C_1;

  bool accepted = false;

  // Computing Runge-Kutta-Fehlberg weights
  V->computeNewElements(C);
```

```
h->computeNewElements(C, t);

k1 = -i*dt*slater->computeRightHandSide(A);
m1 = -i*dt*orbital->computeRightHandSide(C, A);

V->computeNewElements(C + m1/4.0);
h->computeNewElements(C + m1/4.0, t + dt*0.25);

k2 = -i*dt*slater->computeRightHandSide(A + k1/4.0);
m2 = -i*dt*orbital->computeRightHandSide(C + m1/4.0, A +
     k1/4.0);

V->computeNewElements(C + 3.0/32*m1 + 9.0/32*m2);
h->computeNewElements(C + 3.0/32*m1 + 9.0/32*m2, t + dt
    *3.0/8.0);

k3 = -i*dt*slater->computeRightHandSide(A + 3.0/32*k1 +
    9.0/32*k2);
m3 = -i*dt*orbital->computeRightHandSide(C + 3.0/32*m1 +
     9.0/32*m2, A + 3.0/32*k1 + 9.0/32*k2);

V->computeNewElements(C + 1932.0/2197*m1 - 7200.0/2197*
    m2 + 7296.0/2197*m3);
h->computeNewElements(C + 1932.0/2197*m1 - 7200.0/2197*
    m2 + 7296.0/2197*m3, t + dt*12.0/13.0);

k4 = -i*dt*slater->computeRightHandSide(A + 1932.0/2197*
    k1 - 7200.0/2197*k2 + 7296.0/2197*k3);
m4 = -i*dt*orbital->computeRightHandSide(C +
    1932.0/2197*m1 - 7200.0/2197*m2 + 7296.0/2197*m3, A +
     1932.0/2197*k1 - 7200.0/2197*k2 + 7296.0/2197*k3);

V->computeNewElements(C + 439.0/216*m1 - 8.0*m2 +
    3680.0/513*m3 - 845.0/4104*m4);
h->computeNewElements(C + 439.0/216*m1 - 8.0*m2 +
    3680.0/513*m3 - 845.0/4104*m4, t + dt);

k5 = -i*dt*slater->computeRightHandSide(A + 439.0/216*k1
     - 8.0*k2 + 3680.0/513*k3 - 845.0/4104*k4);
m5 = -i*dt*orbital->computeRightHandSide(C + 439.0/216*
    m1 - 8.0*m2 + 3680.0/513*m3 - 845.0/4104*m4, A +
    439.0/216*k1 - 8.0*k2 + 3680.0/513*k3 - 845.0/4104*k4
    );

V->computeNewElements(C - 8.0/27*m1 + 2.0*m2 -
    3544.0/2565*m3 + 1859.0/4104*m4 - 11.0/40*m5);
h->computeNewElements(C - 8.0/27*m1 + 2.0*m2 -
    3544.0/2565*m3 + 1859.0/4104*m4 - 11.0/40*m5, t + dt
    *0.25);
```

```
k6 = -i*dt*slater->computeRightHandSide(A - 8.0/27*k1 +
    2.0*k2 - 3544.0/2565*k3 + 1859.0/4104*k4 - 11.0/40*k5
    );
m6 = -i*dt*orbital->computeRightHandSide(C - 8.0/27*m1 +
     2.0*m2 - 3544.0/2565*m3 + 1859.0/4104*m4 - 11.0/40*
    m5, A - 8.0/27*k1 + 2.0*k2 - 3544.0/2565*k3 +
    1859.0/4104*k4 - 11.0/40*k5);


// Computing new states
A_  = A + 16.0/135*k1 + 6656.0/12825*k3 + 28561.0/56430*
    k4 - 9.0/50*k5 + 2.0/55*k6;
A_1 = A + 25.0/216*k1 + 1408.0/2565*k3 + 2197.0/4104*k4
    - 1.0/5*k5;

C_  = C + 16.0/135*m1 + 6656.0/12825*m3 + 28561.0/56430*
    m4 - 9.0/50*m5 + 2.0/55*m6;
C_1 = C + 25.0/216*m1 + 1408.0/2565*m3 + 2197.0/4104*m4
    - 1.0/5*m5;

// Computing the error
vec R;
double maxR;
double rTmp;

R = 1.0/dt*abs(A_ - A_1);
maxR = max(R);
for(uint i=0; i<C.n_cols; i++){
  R = 1.0/dt*abs(C_.col(i) - C_1.col(i));
  rTmp = max(R);
  if(rTmp > maxR)
  maxR = rTmp;
}

// Computing the time-weight
double delta = 0.84*pow(epsilon/maxR, 0.25);

if(maxR <= epsilon){
  C = C_1;
  A = A_1;
  dt = delta*dt;

  // Re-ormalizing
  renormalize(C);
  A = A/sqrt(cdot(A,A));

  accepted = true;
}else
```

```
  {
    dt = delta*dt;
    step--;
  }

  return accepted;
}
```

Listing 10.3: Implementation of Runge-Kutta-Fehlberg for MCTDHF equations of motion.

## 10.4 Other Integration Methods

There is a multitude of possible integrators to use. The difficulty lies in finding the one that is best for your problem. One scheme of interest is the splitting method [49], a cheap and accurate fourth order integrator. For the implementation of splitting methods in the MCTDHF method, see reference [50]. Another option is the Constant Mean Field [32] integration scheme which has proven itself to work well for the MCTDH equations of motion.

# Chapter 11

# Computing the Mean Fields

In this chapter the computation of the mean fields and the interaction elements are described. We will also show an approximation scheme used to simplify the calculation of the mean fields.

## 11.1 Mean fields and interaction elements

A huge part of the computational cost of solving the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) equations is the computation of the mean fields and the interaction elements. At every time step, both the mean fields and the interaction elements must be recalculated, since the orbitals change at every time step. The mean fields are given by

$$V^{pq}(\mathbf{r}) = \int \phi_p^\dagger(\mathbf{r}')V(\mathbf{r},\mathbf{r}')\phi_q(\mathbf{r}')d\mathbf{r}' \ . \tag{11.1}$$

For every $p$ and $q$ the mean field must be evaluated at every $\mathbf{r}$-coordinate. In our calculations space is represented on a uniform finite grid. At every point in the grid the mean field must be evaluated.

The most convenient way of computing a mean field on a uniform grid, is by using the trapezoidal integration method. The trapezoidal integration algorithm is summarized by

$$\int f(\mathbf{r}_j,\mathbf{r}')d\mathbf{r}' \approx \frac{\Delta\mathbf{r}}{2}\sum_{i=1}^{N-1}\left(f(\mathbf{r}_j,\mathbf{r}'_{i+1}) + f(\mathbf{r}_j,\mathbf{r}'_i)\right) \tag{11.2}$$

$$= \Delta\mathbf{r}\left(\frac{f(\mathbf{r}_j,\mathbf{r}'_1) + f(\mathbf{r}_j,\mathbf{r}'_N)}{2} + \sum_{i=2}^{N-1} f(\mathbf{r}_j,\mathbf{r}'_i)\right) \ , \tag{11.3}$$

on a uniform grid. When we compute the mean fields the $\Delta\mathbf{r}$ is included in the orbitals. Using the trapezoidal integration scheme, the mean field at a point $\mathbf{r}_j$ is approximated by

$$V^{pq}(\mathbf{r}_j) \approx \frac{\phi_p^\dagger(\mathbf{r}'_1)V(\mathbf{r}_j,\mathbf{r}'_1)\phi_q(\mathbf{r}'_1) + \phi_p^\dagger(\mathbf{r}'_N)V(\mathbf{r}_j,\mathbf{r}'_N)\phi_q(\mathbf{r}'_N)}{2} \tag{11.4}$$

$$+ \sum_{i=2}^{N-1} \phi_p^\dagger(\mathbf{r}'_i)V(\mathbf{r}_j,\mathbf{r}'_i)\phi_q(\mathbf{r}'_i) \ . \tag{11.5}$$

## 11.2 Approximation to the mean field operators

Computing the mean field operators directly is an expensive operation. For the MCTDHF method to scale well numerically, the mean fields must be approximated efficiently. There are several schemes that can be used to ease these computations. The most straightforward way is to approximate the interaction matrix by a low rank approximation and combining this with a H-matrix formulation [51]. In the low rank approximation we want to approximate the interaction potential by a sum

$$V(\mathbf{r},\mathbf{r}') \approx \sum_m \lambda_m U_m(\mathbf{r})U_m(\mathbf{r}') \ , \tag{11.6}$$

where we have separated the dependence of $\mathbf{r}$ and $\mathbf{r}'$. To achieve this, the interaction operator is first discretized in a sub-basis of the total discrete space. Then a low rank approximation is performed. For more details, see Refs. [52] and [53].

### 11.2.1 Discretization

The discretization is done by expanding the interaction operator in a discrete basis

$$\hat{V}_{app} = \hat{R}\hat{V}\hat{R} = \sum_{ij}^{L}\sum_{i'j'}^{L} |i\rangle\, Q^{-1}_{ij}\tilde{V}_{jj'}Q^{-1}_{j'i'}\, \langle i'| \ , \tag{11.7}$$

where $R$ is the projection operator given by

$$\hat{R} = \sum_{ij} |i\rangle\, Q^{-1}_{ij}\, \langle j| \ , \tag{11.8}$$

$$Q_{ij} = \langle i|j\rangle = \int \langle i|\mathbf{r}\rangle\langle\mathbf{r}|j\rangle \ d\mathbf{r} \ , \tag{11.9}$$

and the interaction matrix elements are given by

$$\tilde{V}_{ij} = \langle i|V|j\rangle = \int\int \langle i|\mathbf{r}\rangle V(\mathbf{r},\mathbf{r}')\langle \mathbf{r}'|j\rangle \, d\mathbf{r}d\mathbf{r}' = \int\int h_i(\mathbf{r})V(\mathbf{r},\mathbf{r}')h_j(\mathbf{r}') \, d\mathbf{r}d\mathbf{r}' \, . \tag{11.10}$$

### 11.2.2 Low rank approximation

To efficiently calculate the matrix elements of the interaction operator, we want to write it as a product of single-particle function. To control the local error we introduce the weight overlap matrix

$$S_{ij} = \int g(\mathbf{r})h_i(\mathbf{r})h_j(\mathbf{r}) \, d\mathbf{r} \, , \tag{11.11}$$

where the weight function $g(\mathbf{r}) > 0$. The $\mathbf{S}$-matrix is positive definite and symmetric. We can now set up the generalized eigenvalue problem

$$\tilde{\mathbf{V}}\mathbf{u}_m = \lambda_m \mathbf{S}\mathbf{u}_m \, . \tag{11.12}$$

To solve this we perform a Cholesky factorization of the $\mathbf{S}$-matrix:

$$\mathbf{S} = \tilde{\mathbf{C}}^T\tilde{\mathbf{C}} \, , \tag{11.13}$$

where $\mathbf{C}$ is lower triangular. The eigenvalue problem can now be written as

$$\tilde{\mathbf{V}}\mathbf{u}_m = \lambda_m \tilde{\mathbf{C}}^T\tilde{\mathbf{C}}\mathbf{u}_m \, . \tag{11.14}$$

Some reorganization of the equation gives us

$$\left(\tilde{\mathbf{C}}^T\right)^{-1}\tilde{\mathbf{V}}\left(\tilde{\mathbf{C}}\right)^{-1}\tilde{\mathbf{C}}\mathbf{u}_m = \lambda_m \tilde{\mathbf{C}}\mathbf{u}_m \, . \tag{11.15}$$

Defining $\tilde{\mathbf{C}}\mathbf{u}_m = \tilde{\mathbf{u}}_m$ gives us an ordinary eigenvalue problem

$$\left(\tilde{\mathbf{C}}^T\right)^{-1}\tilde{\mathbf{V}}\left(\tilde{\mathbf{C}}\right)^{-1}\tilde{\mathbf{u}}_m = \lambda_m \tilde{\mathbf{u}}_m \, . \tag{11.16}$$

A spectral decomposition of the problem yields

$$\left(\tilde{\mathbf{C}}^T\right)^{-1}\tilde{\mathbf{V}}\left(\tilde{\mathbf{C}}\right)^{-1} = \tilde{\mathbf{U}}\Lambda\tilde{\mathbf{U}}^T \quad \Rightarrow \quad \tilde{\mathbf{V}} = \tilde{\mathbf{C}}^T\tilde{\mathbf{U}}\mathbf{\Lambda}\tilde{\mathbf{U}}^T\tilde{\mathbf{C}} \, . \tag{11.17}$$

We can now write the interaction matrix as

$$\tilde{\mathbf{V}} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T \, , \tag{11.18}$$

where $\mathbf{U} = \tilde{\mathbf{C}}^T\tilde{\mathbf{U}}$ and $\mathbf{\Lambda}$ is the diagonal matrix filled with eigenvalues. If we now set a threshold $\varepsilon$ so that if $|\lambda_i| < \varepsilon$, $\lambda_i$ is set to zero. Using this approximation the interaction matrix can be written as

$$\tilde{\mathbf{V}}^{\varepsilon} = \mathbf{U}\mathbf{\Lambda}^{\varepsilon}\mathbf{U}^T \ , \tag{11.19}$$

with $\mathbf{\Lambda}^{\varepsilon} = \mathrm{diag}(\lambda_1, \cdots, \lambda_M, 0, \cdots, 0)$. Inserting this approximation into Eq. (11.7) gives us

$$\hat{V}_{app} \approx \hat{V}_{low} = \sum_{ij}^{L}\sum_{i'j'}^{L} |i\rangle\, Q^{-1}_{ij}\tilde{V}^{\varepsilon}_{jj'}Q^{-1}_{j'i'}\, \langle i'| \tag{11.20}$$

$$= \sum_{m}\sum_{ij}^{L}\sum_{i'j'}^{L} |i\rangle\, Q^{-1}_{ij}U_{jm}\lambda_m U_{mj'}Q^{-1}_{j'i'}\, \langle i'| \tag{11.21}$$

$$= \sum_{m}\lambda_m \sum_{i}^{L} |i\rangle\, [\mathbf{Q}^{-1}\mathbf{U}]_{im} \sum_{i'}^{L}[\mathbf{U}\mathbf{Q}^{-1}]_{mi'}\, \langle i'| \ . \tag{11.22}$$

In spatial coordinates:

$$\hat{V}_{low}(\mathbf{r},\mathbf{r}') = \sum_{m}\lambda_m \sum_{ii'}^{L} h_i(\mathbf{r})[\mathbf{Q}^{-1}\mathbf{U}]_{im}[\mathbf{U}\mathbf{Q}^{-1}]_{mi'}h_{i'}(\mathbf{r}') \ . \tag{11.23}$$

$$\tag{11.24}$$

Simplifying the notation by defining

$$U_m(\mathbf{r}) = \sum_{i}^{L} h_i(\mathbf{r})[\mathbf{Q}^{-1}\mathbf{U}]_{im} \ , \tag{11.25}$$

gives us an equation on the product form:

$$\hat{V}_{low}(\mathbf{r},\mathbf{r}') = \sum_{m}\lambda_m U_m(\mathbf{r})U_m(\mathbf{r}') \ . \tag{11.26}$$

### 11.2.3 Error

Koch has shown [53] that the error of the low rank approximation is

$$\left|\langle\phi_p|\hat{V}_{app} - \hat{V}_{low}|\phi_q\rangle\right| = \mathcal{O}(\varepsilon) \ , \tag{11.27}$$

where $\varepsilon$ is the threshold set for the eigenvalues.

### 11.2.4 Mean Fields and Interaction Elements

The mean fields can be written as

$$V^{qr}(\mathbf{y}_i) = \sum_m \lambda_m U_m(\mathbf{y}_i) \int \phi_q^\dagger(\mathbf{x}) U_m(\mathbf{x}) \phi_r(\mathbf{x}) d\mathbf{x} \tag{11.28}$$

$$= \sum_m \lambda_m U_{im} \sum_{j=1}^{N_{grid}} C_{jq}^\dagger U_{jm} C_{jr} \ . \tag{11.29}$$

The number of operations for all the mean field matrix elements are of order $\mathcal{O}(N_{grid} \times M \times N^2)$, where $M$ is the number of eigenvalues and $N$ the number of orbitals. In contrast; the straight forward integral is of order $\mathcal{O}(N_{grid}^2 \times N^2)$. The interaction elements are given by

$$\langle \phi_p \phi_q | \hat{V} | \phi_r \phi_s \rangle = \sum_m^M \lambda_m \int \phi_p^\dagger(\mathbf{r}) U_m(\mathbf{r}) \phi_r(\mathbf{r}) d\mathbf{r} \int \phi_q^\dagger(\mathbf{r}') U_m(\mathbf{r}') \phi_s(\mathbf{r}') d\mathbf{r}' \tag{11.30}$$

$$= \sum_m^M \lambda_m Z_{m,pr} Z_{m,qs} \ , \tag{11.31}$$

where

$$Z_{m,pr} = \int \phi_p^\dagger(\mathbf{r}) U_m(\mathbf{r}) \phi_r(\mathbf{r}) d\mathbf{r} \ . \tag{11.32}$$

If we store all the $Z$-matrices, the number of operations are of the order $\mathcal{O}(M)$.

### 11.2.5 A Simple Test Implementation

Let us perform a simple test implementation of the low rank approximation. As a basis we choose the same basis used in the spatial discretization of the system, with $N$ grid points. We set $L = N$ and $|i\rangle = |x_i\rangle$. The mass matrix $\mathbf{Q}$ elements are

$$Q_{ij} = \langle x_i | x_j \rangle = \delta_{ij} \ , \tag{11.33}$$

$$\Rightarrow \mathbf{Q} = \mathbb{1} \ . \tag{11.34}$$

and the position representation of the basis is

$$h_i(x) = \delta(x - x_i) \ . \tag{11.35}$$

For the interaction potential we use a shielded Coulomb interaction

$$V(x,y) = \frac{1}{\sqrt{(x-y)^2 + a^2}} \ . \tag{11.36}$$

The weight function $g(x)$ is set to be constant 1 for $|x| \in [0,2]$ and to decrease linearly from 1 to 0.1 for $|x| \in (2,10]$. The weighted overlap matrix $\mathbf{S}$ is then

$$S_{ij} = g(x_i)\delta_{ij} \ . \tag{11.37}$$

A quick summary of the implementation of the algorithm:

1. Choose a basis $\{|i\rangle\}$ and discretize it in an appropriate subspace of the position space to get $h_i(x)$.
2. Set up the $\mathbf{Q}$-matrix
3. Discretize the interaction matrix $\tilde{\mathbf{V}}$
4. Calculate the $\mathbf{S}$-matrix
5. Perform a Cholesky decomposition of the $\mathbf{S}$ to find $\tilde{\mathbf{C}}$
6. Find the eigenvalues and eigenvectors of $\left(\tilde{\mathbf{C}}^T\right)^{-1} \tilde{\mathbf{V}} \left(\tilde{\mathbf{C}}\right)^{-1}$
7. Compute $\mathbf{U} = \tilde{\mathbf{C}}^T \tilde{\mathbf{U}}$
8. Find all the $U_m$ elements:

$$U_m(\mathbf{r}) = \sum_i^L h_i(\mathbf{r}) [\mathbf{Q}^{-1}\mathbf{U}]_{im} \ , \tag{11.38}$$

Figure 11.1 shows the result of this test implementation. Using only 110 of the original 200 elements the maximal relative error is about 0.0009.

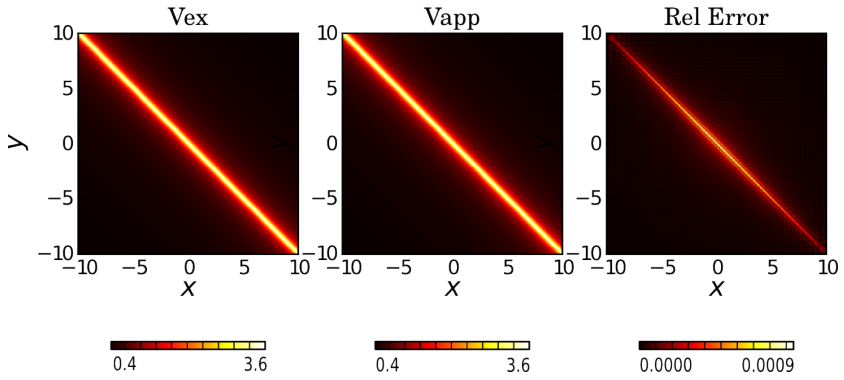



Figure 11.1: The figure on the left shows the shielded Coulomb potential. The figure in the center shows the low rank approximation to the potential, using $\varepsilon = 1$. The figure on the right shows the relative error.

# Chapter 12

# Validation

An important part of software-development is validation of the code. Code validation is the process of verification of the code so that we know it produced the correct results.

The code validation process is started by comparing results with a special case where an analytic solution can be found. After this verification the program is tested against a study by Zanghellini *et al* [54]. In this study the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) method applied to a one dimensional quantum dot. First the ground state energy is calculated using imaginary time propagation. Using the ground state as a starting point an electric field simulating a laser is applied to the system. The results are compared and the implementation is validated.

## 12.1 An analytic comparison

As a first validation of the code, we will construct a simple test case for two spin half particles confined in one dimension. The particles are confined in a harmonic oscillator potential

$$V_{conf} = \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 \ , \tag{12.1}$$

where $x_1$ is the coordinate of the first particle and $x_2$ of the second. The interaction between the particles is modeled by

$$V(x_1, x_2) = -\varepsilon|x_1 - x_2|^2 \ , \tag{12.2}$$

where $\varepsilon$ is the interaction strength. This problem can be solved analytically, making it a perfect test for our numerical implementation. The total

Hamiltonian reads

$$H = -\frac{1}{2}\frac{\partial^2}{\partial x_1^2} - \frac{1}{2}\frac{\partial^2}{\partial x_2^2} + \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 - \varepsilon|x_1 - x_2|^2 \ . \tag{12.3}$$

### 12.1.1 Analytic solution

To solve the problem analytically we have to rewrite the Hamiltonian by performing a change of variables. Let us start by defining

$$R = \frac{1}{\sqrt{2}}(x_1 + x_2) \ , \quad r = \frac{1}{\sqrt{2}}(x_1 - x_2) \ . \tag{12.4}$$

Using these relations, we can find

$$R^2 + r^2 = x_1^2 + x_2^2 \tag{12.5}$$

$$\frac{\partial^2}{\partial R^2} + \frac{\partial^2}{\partial r^2} = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} \tag{12.6}$$

$$|x_1 - x_2|^2 = 2r^2 \ . \tag{12.7}$$

The Hamiltonian can now be written as

$$\hat{H} = -\frac{1}{2}\frac{\partial^2}{\partial R^2} - \frac{1}{2}\frac{\partial^2}{\partial r^2} + \frac{1}{2}R^2 + \frac{1}{2}r^2 - 2\varepsilon r^2 \tag{12.8}$$

$$= \hat{H}_R + \hat{H}_r \ . \tag{12.9}$$

The hamiltonian is now separable. The two solutions are given by

$$\hat{H}_R\phi(R) = \epsilon_R \tag{12.10}$$

and

$$\hat{H}_r\psi(r) = \epsilon_r \ , \tag{12.11}$$

where the total energy is $E = \epsilon_R + \epsilon_r$. Solving for $R$ we get the standard Harmonic oscillator solution

$$\left(-\frac{1}{2}\frac{\partial^2}{\partial R^2} + \frac{1}{2}R^2\right)\phi(r) = \epsilon_R\phi(R) \ , \tag{12.12}$$

with the energy being

$$\epsilon_R = n_1 + \frac{1}{2}, \tag{12.13}$$

where $n_1 = 0, 1, \cdots$ . Now for $r$:

$$\left(-\frac{1}{2}\frac{\partial^2}{\partial r^2} + \frac{1}{2}\left(1 - 4\varepsilon\right)r^2\right)\psi(r) = \epsilon_r\psi(r) \ . \tag{12.14}$$

This is also the harmonic oscillator problem, and the solution is

$$\epsilon_r = \sqrt{1-4\varepsilon}\left(n_2 + \frac{1}{2}\right) \ , \tag{12.15}$$

where $n_2 = 0, 1, \cdots$ . The total energy is given by

$$E = \frac{1}{2} + n_1 + \sqrt{1-4\varepsilon}\left(n_2 + \frac{1}{2}\right) \ . \tag{12.16}$$

Figure 12.1 shows the lowest energies as a function of $\epsilon$ for the closed form expression.

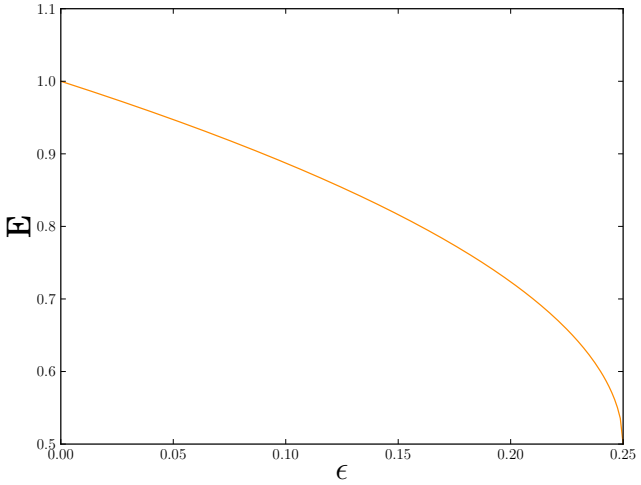



Figure 12.1: The analytical solution of the energy as a function of $\varepsilon$ for a constructed system where the particles interact via an oscillator potential where $\varepsilon$ is the interaction strength.

### 12.1.2 Numerical Solution

To solve the oscillator potential numerically the oscillator interaction was first implemented into the program as a subclass of the interaction class. The spatial grid is uniformly distributed on the domain $\Gamma = [-10, 10]$ using 128 grid points. The differential operator is calculated using Fourier transformations. Figure 12.2 shows computations performed for different values of $\varepsilon$ together with the exact solution. The numerical error is also shown, and it is calculated by taking the absolute difference between the numerical calculations and the closed form expression. The figure shows us that the numerical

solution is very close to the analytic result for a low interaction strength $\varepsilon$. Increasing the interaction strength creates a larger difference between the results of the numerical simulation and the analytic result, as is expected for a stronger interacting system. Increasing the number of orbitals makes the program able to handle higher degrees of correlation, and it will produce results closer to the analytical result. The convergence of the imaginary time propagation is faster at lower $\varepsilon$. This is due to the initial guess being closer to the actual solution.

## 12.2 Replicating a Study of a Two-Electron Quantum Dot

A study by Zanghellini *et al* [54] tests the MCTDHF method for a two-electron quantum dot in one spatial dimension. The electrons are confined by a harmonic oscillator potential and the electron-electron interaction is modeled using a shielded Coulomb potential. The ground state is found by performing an imaginary time propagation. Using the ground state as a starting point, the system is radiated by a laser for a time-dependent study.

The electrons are confined by a harmonic oscillator potential

$$V_{conf}(x_1, x_2) = \frac{\Omega^2}{2}(x_1 + x_2) \ , \tag{12.17}$$

where $\Omega$ is the frequency of the harmonic oscillator. The interaction between the electrons is modeled by a shielded Coulomb potential

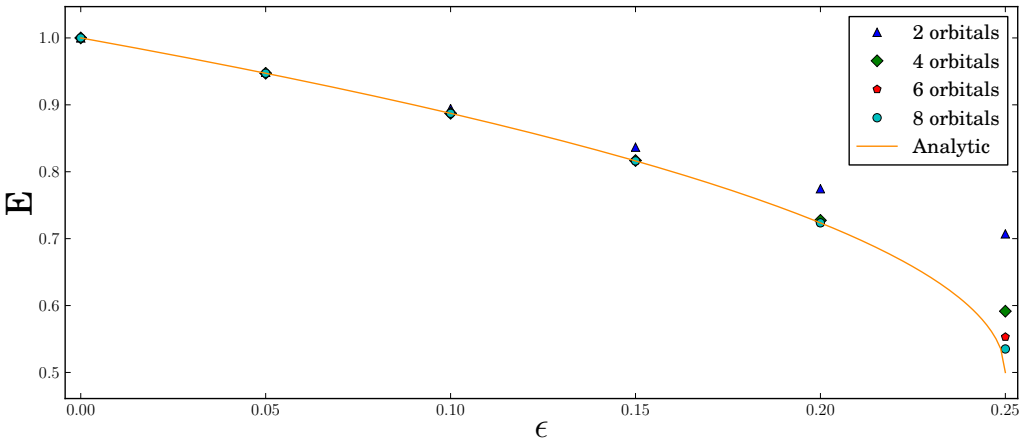
$$V_I(x_1, x_2) = \frac{1}{\sqrt{(x_1 - x_2)^2 + a^2}} \ , \tag{12.18}$$

where $a$ is a shielding parameter. The full Hamiltonian describing the system is

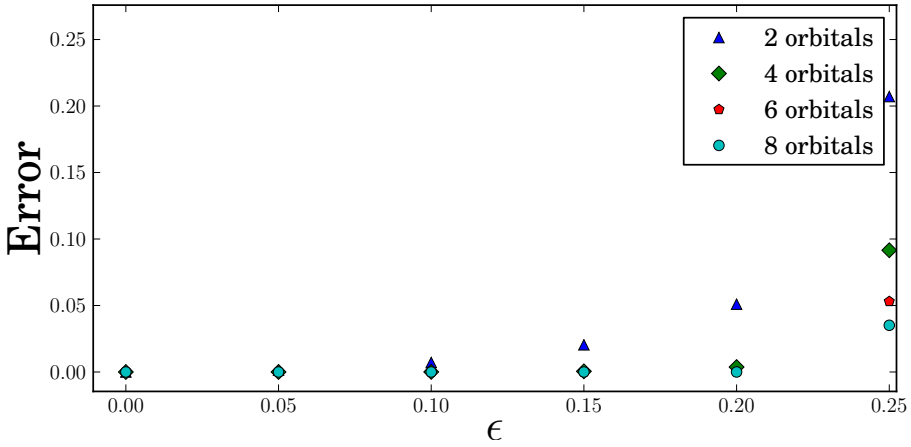
$$H(x_1, x_2) = \frac{p_1^2}{2} + \frac{p_2^2}{2} + \frac{\Omega^2}{2}(x_1 + x_2) + \frac{1}{\sqrt{(x_1 - x_2)^2 + a^2}} \ . \tag{12.19}$$

The laser used in the time-dependent study is modeled by

$$V_{laser}(x_1, x_2, t) = (x_1 + x_2)\sin(\omega t) \ . \tag{12.20}$$




(a) A comparison of the energies computed numerically and the exact energy given by the analytic expression in Eq (12.16).




(b) Absolute errors in the numerical computations. The error is given by the absolute difference of the numerical results to the analytic solution.

Figure 12.2: A comparison of the numerical and the analytic results for the oscillator interaction potential, where $\varepsilon$ is the strength of the interaction. The computations are performed for different different sizes of the orbital-basis.

### 12.2.1 Measuring quantities of interest

#### Degree of correlation

The degree of correlation[1], $K$, is defined as

$$K(t) = \sum_i \frac{1}{|p_i(t)|^4} \ , \tag{12.21}$$

where $p_i$ are diagonal elements of the one-body density $\rho$.

#### Overlap with the ground state

To monitor the behavior of the time-dependent system we observe the overlap with the ground state $|\langle\Psi(0)|\Psi(t)\rangle|^2$. The overlap is given by

$$\langle\Psi(0)|\Psi(t)\rangle = \sum_{\alpha,\beta} A^\dagger_\alpha(0) A_\beta(t) \langle\Phi_\alpha(0)|\Phi_\beta(t)\rangle \ . \tag{12.22}$$

For two electrons the explicit form of the overlap is

$$\langle\Phi_\alpha(0)|\Phi_\beta(t)\rangle = \langle\phi_{\alpha_1}(0)|\phi_{\beta_1}(t)\rangle\langle\phi_{\alpha_2}(0)|\phi_{\beta_2}(t)\rangle - \langle\phi_{\alpha_1}(0)|\phi_{\beta_2}(t)\rangle\langle\phi_{\alpha_2}(0)|\phi_{\beta_1}(t)\rangle \ . \tag{12.23}$$

## 12.3 Results

To reproduce Zanghellini's results, we have taken care to prepared our calualtions as similarly as possible to the systems described in [54]. The grid is uniform on the domain $\Gamma = [-10, 10]$, using a grid spacing of 0.1. The oscillator strength of the confining potential is set to $\omega = 0.25$, and the shielding parameter in the Coulomb interaction is $a = 0.25$. The differential operators are computed using a three-point finite difference scheme and with Fourier transforms. The imaginary and time propagation is performed using a Runge-Kutta 4 and Runge-Kutta-Fehlberg integrator . For imaginary time the optimal time-step varied quite a lot and the Runge-Kutta-Fehlberg integrator was by far the best choice due to the adaptive step length. In the case of ordinary time propagation the time step was approximately constant, and thus there was no need to use the Runge-Kutta-Fehlberg integrator. Table 12.2 shows a comparison between the ground state energies computed by Zanghellini and the ones computed in this thesis. To show the effects of discretization and choice of differential operator, results using Fourier transformations are also presented. The results are in most cases in agreement,

[1] See Section 3.4.

with only minor deviations. Such deviations are to be expected as there might be minor differences in the implementation and in how the system is set up. However, there is a huge difference in the results using four orbitals, where Zanghellini's result is significantly higher. A possible explanation is that the convergence criteria used by Zanghellini was not strong enough. The energy computed during imaginary time propagation can stabilize on plateaus, appearing to have converged, but has in fact not, and a strict convergence criterion is needed. In Figure 12.3 the electron density is presented. A visual overview of the energy convergence, evolution of the orbitals, and the wavefunction coefficients is presented in Figure 12.5.

The time-dependent analysis is performed using the ground state by imaginary time propagation as an initial state. The angular frequency of the laser is set to $\Omega = 8.0$, and the initial state is propagated forward in time. Figure 12.4 shows the overlap of the propagated wavefunction with the initial state. A visual comparison with Zanghellini's time-dependent results coincide well.

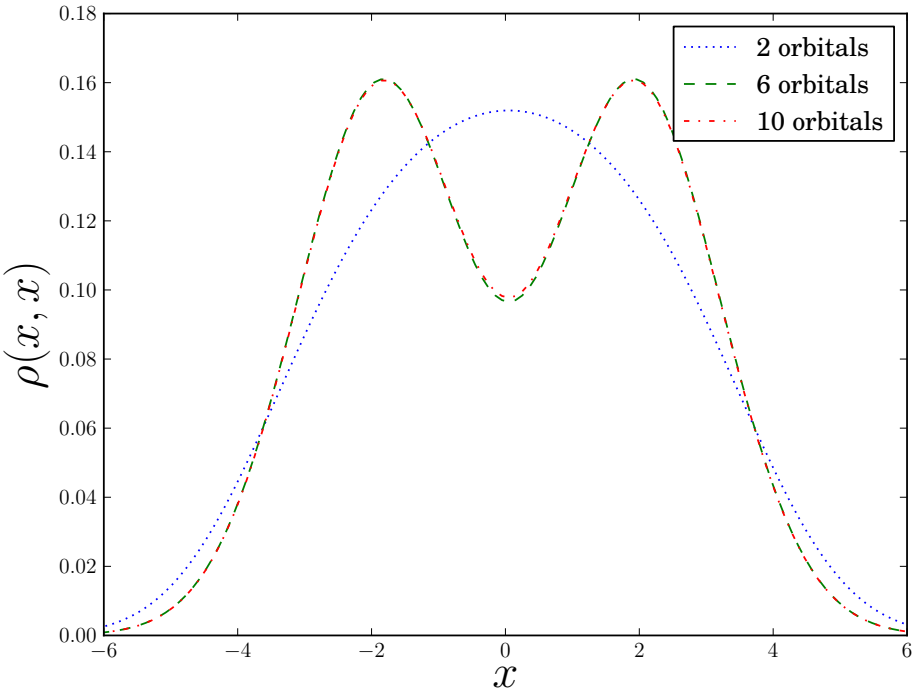



Figure 12.3: One-body density for the one-dimensional quantum dot shown for an increasing number orbitals spanning the single-particle basis. The plot is a replication of Zanghellini's study.

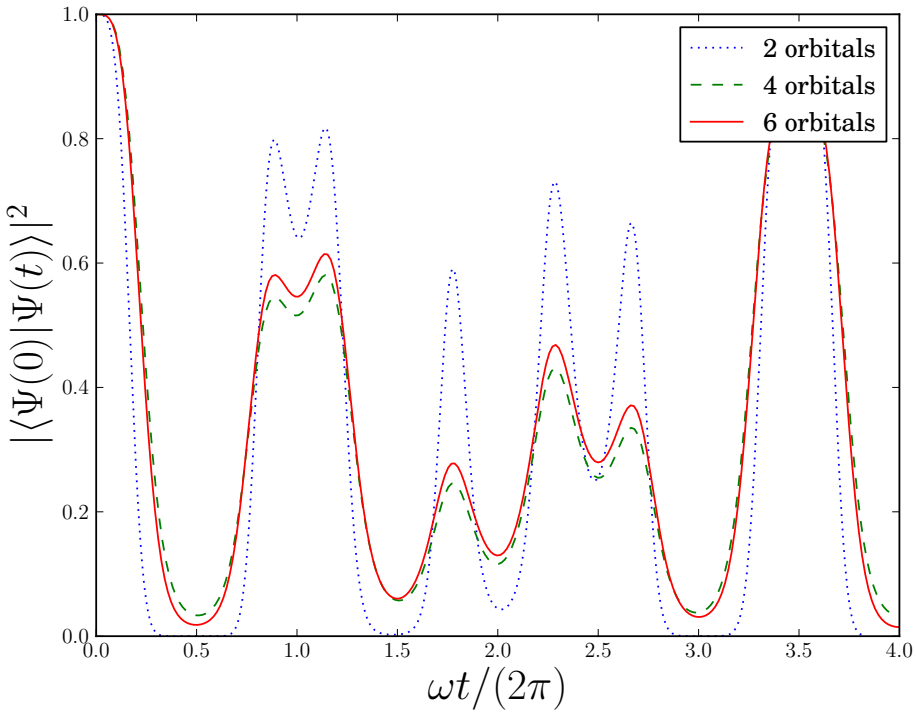


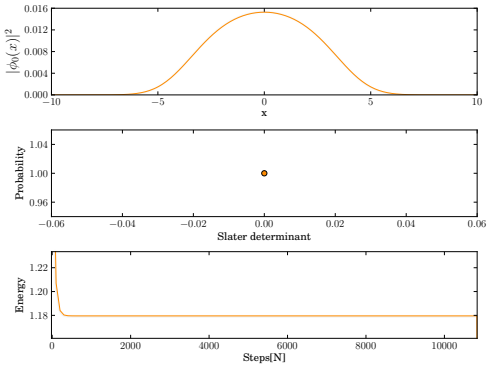

Figure 12.4: Overlap of the intial and the time evolved state for the one-dimensional quantum dot radiated by a laser with angular frequency $\Omega = 8.0$. The results are shown for an increasing number orbitals spanning the single-particle basis. The plot is a replication of Zanghellini's study.

| Orbitals | $E[Ha]$ | $K$ | $\Delta E$ | $E_{\text{Zanghellini}}[Ha]$ | $K_{\text{Zanghellini}}$ |
|---|---|---|---|---|---|
| 2 | 1.179569 | 1.000000 | $10^{-15}$ | 1.1795 | 1.0000 |
| 4 | 0.844959 | 2.000000 | $10^{-15}$ | 1.0214 | 1.3568 |
| 6 | 0.826123 | 1.726119 | $10^{-14}$ | 0.8261 | 1.7260 |
| 8 | 0.825462 | 1.711119 | $10^{-14}$ | 0.8255 | 1.7150 |
| 10 | 0.825031 | 1.702987 | $10^{-14}$ | 0.8250 | 1.7029 |
| 12 | 0.824916 | 1.699905 | $10^{-16}$ | 0.8249 | 1.6997 |

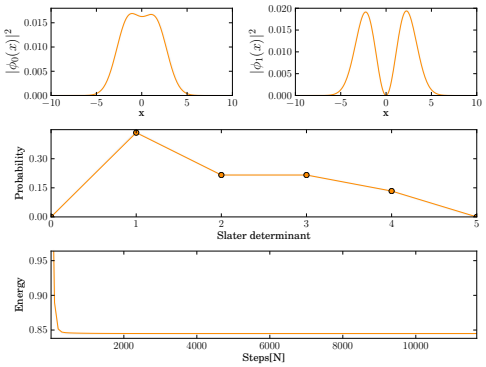
Table 12.1: Results using a three-point finite difference scheme to calculate the kinetic energy. The energy $E$ is measured in Hartrees[Ha] and $K$ is the degree of correlation. The difference in energy, $\Delta E$, is taken between the last sample and the one 100 steps before, and is used as a measure of convergence. The results are shown for an increasing number of orbitals spanning the single-particle basis

| Orbitals | $E[Ha]$ | $K$ | $\Delta E$ | $E_{\text{Zanghellini}}[Ha]$ | $K_{\text{Zanghellini}}$ |
|---|---|---|---|---|---|
| 2 | 1.179585 | 1.000000 | $10^{-15}$ | 1.1795 | 1.0000 |
| 4 | 0.845038 | 2.000000 | $10^{-16}$ | 1.0214 | 1.3568 |
| 6 | 0.826220 | 1.725933 | $10^{-16}$ | 0.8261 | 1.7260 |
| 8 | 0.825559 | 1.710936 | $10^{-15}$ | 0.8255 | 1.7150 |
| 10 | 0.825128 | 1.702811 | $10^{-15}$ | 0.8250 | 1.7029 |

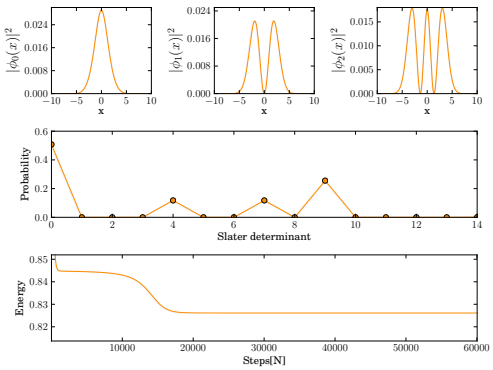
Table 12.2: Results using Fourier transformations to calculate the kinetic energy. The energy $E$ is measured in Hartrees[Ha] and $K$ is the degree of correlation. The difference in energy, $\Delta E$, is taken between the last sample and the one 100 steps before, and is used as a measure of convergence. The results are shown for an increasing number of orbitals spanning the single-particle basis



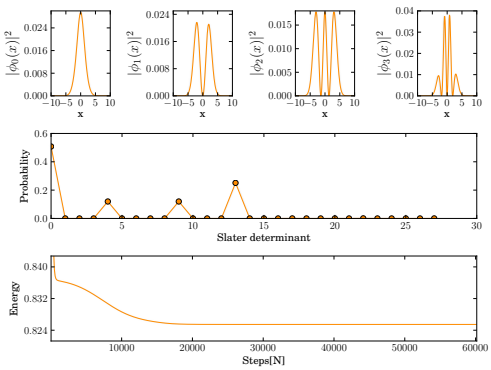

(a) Convergence for 1 spatial orbital.



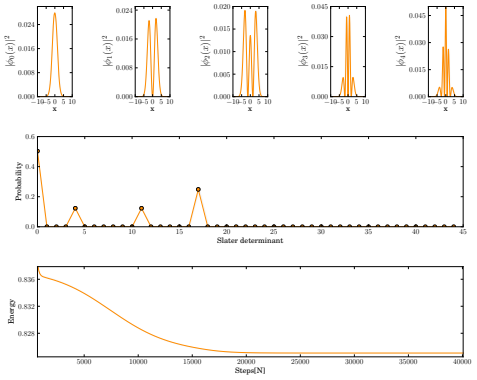

(b) Convergence for 2 spatial orbitals.




(c) Convergence for 3 spatial orbitals.




(d) Convergence for 4 spatial orbitals.




(e) Convergence for 5 spatial orbitals.

Figure 12.5: The plots shows the spatial projection of the spin-independent one-particle orbitals, the probability weighting of the expansion coefficients of the full wavefunction (in a Slater determinant expansion) and the energy convergence for the one-dimensional quantum dot.

# Part III

# Systems and Results

# Chapter 13

# Systems

This chapter introduces the potentials we will use in our studies of the quantum dots. Where applicable a physical interpretation is also given. To explore how the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF) method works, a variety of potentials are tested to analyze its strengths and weaknesses. We also want to show the flexibility of the numerical implementation. The system of choice are the quantum dots, i.e., electrons confined by some external potential. We will model the single quantum dot using a harmonic oscillator potential, in one- and two-dimensions. The ground state energies are well-known for this system [55] and for closed shell systems of up to 20 electrons [56] and higher, but for time-dependence little has been done. For a time-dependent study an electric field representing a laser is applied to the system. Other systems of interest are the double quantum dots, which can be used to simulate, e.g., qubits in quantum computers [57]. The parameterization we use for the double well potential has a finite nature and can be tuned to produce results closer to experiments.

## 13.1 Quantum Dots

Quantum dots are electrons confined in semiconducting heterogeneous structures and are typically on the nanometer scale. Typical quantum behavior is shown through discrete levels of energy, and similar to atoms there are magic numbers of electrons exhibiting higher binding energies. Due to similarities with atoms quantum dots are often called *designer atoms*, as their basic properties can be tweaked by modifying the confining potential, while they retain their atomic like properties, even on the nanometer scale. In recent years the interest in quantum dots have attracted lots of attention, with possible uses in medial imaging [58], solar cells [59], quantum computers [60],

and high resolution displays [61] to name a few areas of research.

For a thorough description of the electronic structure of quantum dots the reader is referred to [1].

## 13.2 The model Hamiltonian

The general form of the Hamiltonian used is

$$H = \sum_{i=1}^{N} \left( \frac{1}{2}[\mathbf{p}^2 + \mathbf{A}(t)]^2 + V_{conf}(\mathbf{r}_i) + \sum_{j<i}^{N} V_I(\mathbf{r}_i, \mathbf{r}_j) \right) , \tag{13.1}$$

where $\mathbf{p}$ is the momentum, $\mathbf{A}$ is the electric vector potential, $V_{conf}(\mathbf{r}_i)$ is the one-body confining potential acting on particle $i$, $V_I(\mathbf{r}_i, \mathbf{r}_j)$ is the interaction between particle $i$ and $j$. In all our calculations the interaction potential will be modeled by the shielded Coulomb potential.

Using the *length gauge* and the *dipole approximation* the Hamiltonian can be written as

$$H = \sum_{i=1}^{N} \left( \frac{1}{2}\mathbf{p}^2 + V_{conf}(\mathbf{r}_i) + \sum_{j<i}^{N} V_I(\mathbf{r}_i, \mathbf{r}_j) + \mathbf{E}(t) \cdot \mathbf{r}_i \right) , \tag{13.2}$$

where $\mathbf{E}$ is a time-dependent electric field. In all our calculations we will use a dimensionless form of this Hamiltonian.

### 13.2.1 The Interaction Potential

For the interaction between the electrons we will use a shielded Coulomb interaction [62]

$$V_I(\mathbf{r}_1, \mathbf{r}_2) = \frac{\lambda}{\sqrt{|\mathbf{r}_i - \mathbf{r}_j|^2 + a^2}} , \tag{13.3}$$

where and $a$ is a shielding parameter and $\lambda$ is the effective strength of the interaction given by

$$\lambda = \frac{e^2}{4\pi\epsilon_0\epsilon_r} , \tag{13.4}$$

where $\epsilon_0$ is the permittivity in free space, $\epsilon_r$ is the relative permittivity, and $e$ is the electron charge. If we are using dimensionless units[1], $\lambda$ is written in a dimensionless form:

$$\overline{\lambda} = \frac{e^2}{V_0 L_0 4\pi\epsilon_0\epsilon_r} , \tag{13.5}$$

[1] See section 3.5.

where $L_0$ is the natural length scale and $V_0$ is a scaling factor. The shielded Coulomb interaction is used to avoid singularities arising from the fact that we are using a finite grid where $|\mathbf{r}_i - \mathbf{r}_j|^2$ will be zero when $\mathbf{r}_i = \mathbf{r}_j$. The physical interpretation of the shielding parameter is the freedom the electrons have in moving in a 'confined' direction.

### 13.2.2 Electromagnetic Fields

For systems where the Hamiltonian is described in the dipole approximation and the length gauge, we can simulate a laser [63] with the electromagnetic field:

$$\mathbf{E} = E_0 \sin(\Omega t)\hat{i}_\epsilon. \tag{13.6}$$

where $\hat{i}_\epsilon$ is the polarization vector, $\Omega$ is the angular frequency, and $E_0$ is the amplitude. The polarization vector is usually chosen as one of the cardinal axis of the system, i.e., the $x$-axis of the system. For a dimensionless form the amplitude $E_o$ is divided by the scaling factor $V_0$.

## 13.3 Confining Potentials

A quantum dot is an external potential that confines electrons within a spatial region. We will now describe some forms of this confining potential.

### 13.3.1 The Quantum Dot

We will now look into the single quantum dot systems. This is mostly an academic model, making it perfect for comparison as the ground state energies are well known. We will start by performing an imaginary time propagation to find the ground state. After the ground state is found, the system, with the initial state being the ground state, is then radiated by a laser. Different frequencies of the laser is tested on the system, to measure how the system reacts.

#### Confining Potential

The confining potential is modeled as a harmonic oscillator potential

$$V_{conf}(\mathbf{r}_1, \cdots, \mathbf{r}_N) = \frac{\omega^2}{2} \sum_{i=1}^{N} r_i^2 \ . \tag{13.7}$$

The same potential is used for one-dimensional and two-dimensional computations. Figure (13.1) illustrates the form of the potential.

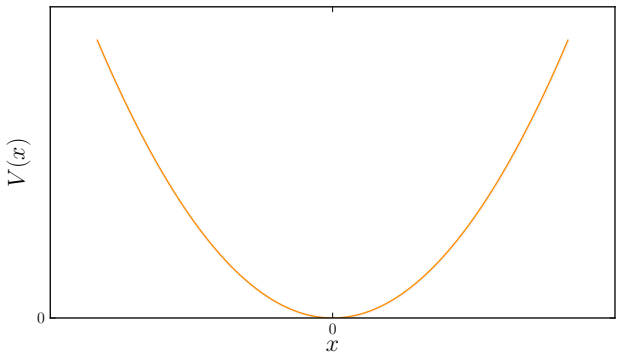



Figure 13.1: An illustration of the one-dimensional harmonic well potential as a function of the distance $x$.

### Time-Evolution

To study the dynamics of a quantum dot, we will apply a simple time-dependent electric field

$$V_l(\mathbf{r}_1, \cdots, \mathbf{r}_N) = \sum_i \mathbf{E}(t) \cdot \mathbf{r}_i = E_0 \sin(\Omega t) \sum_i x_i \ , \tag{13.8}$$

simulating a laser with a polarization axis along the $x$-axis of the system. The energy of such a laser is found using the *Planck relation*, relating energy and frequency. The relation is

$$E = h\nu \tag{13.9}$$

where $\nu$ is the frequency, or using the angular frequency:

$$E = \hbar\tilde{\Omega} \ . \tag{13.10}$$

To excite a system, the quantized difference between two energy levels, $\Delta E$, must be applied to the system. One way of applying this energy is by radiating the system with a laser. If the angular frequency of the laser is decided by Eq (13.10), that is $\Omega = \frac{\Delta E}{\hbar}$ , and the system is radiated by a short burst from the laser using this angular frequency, the result should be an excitation of the system.

## 13.3.2 The Double Quantum Dot

We will now look at the double quantum dot, or the so called quantum dot molecule. The double quantum dot is basically two potential wells in

close vicinity of one another. Such a systems can be used to simulate, for example, the two-qubit quantum gate in the atom-chip configuration[64]. The quantum dots are commonly made of gallium arsenide (GaAs) [13], with typical parameters being

$$m* \simeq 0.067 \ , \quad \varepsilon_r \simeq 12.4 \ ,$$

where $m^*$ is the effective mass and $\varepsilon_r$ is the relative permittivity.

### The One-Dimensional Double Quantum Dot

The double well potential is modeled after the one used by Kryvi in his master thesis [12]:

$$V(\mathbf{r}) = \frac{1}{2d^2}\left(r - \frac{d}{2}\right)^2\left(r + \frac{d}{2}\right)^2 \ , \tag{13.11}$$

where $d$ is the distance between the center of the wells.

### The Two-Dimensional Double Quantum Dot

The potential [57] we use to model the two-dimensional double quantum dot is a linear combination of three Gaussians, and has the form

$$\begin{aligned} V(\mathbf{r}) = &\left[V_a \exp\left(-\frac{(x-a)^2}{l_x^2}\right) + V_b \exp\left(-\frac{(x+a)^2}{l_x^2}\right)\right]\exp\left(-\frac{y^2}{l_y^2}\right) \\ &+ V_c \exp\left(-\frac{x^2}{l_{cx}^2}\right)\exp\left(-\frac{y^2}{l_{cy}^2}\right) \ . \end{aligned} \tag{13.12}$$

An illustration of the form of the potential is shown in figure 13.2. Typical parameter sizes[57] are

$$a = 30 \text{ nm}, \quad V_c = [20, 25, 30] \text{ meV}, \quad V_a = V_b = V_0 = 60 \text{ meV} \ .$$

For a dimensionless scaling, $V_0$, is used as the scaling parameter.

### Time Evolution

The basic time evolution scheme will will employ is to radiate the double quantum dot with a laser, and observe how this affects the electron density. Another, approach is to simulate qubits by applying an external magnetic field. However, by applying a magnetic field we can no longer take advantage of the spin symmetries used to optimize the code, and therefore the application of magnetic fields is outside the scope of this thesis. Another interesting

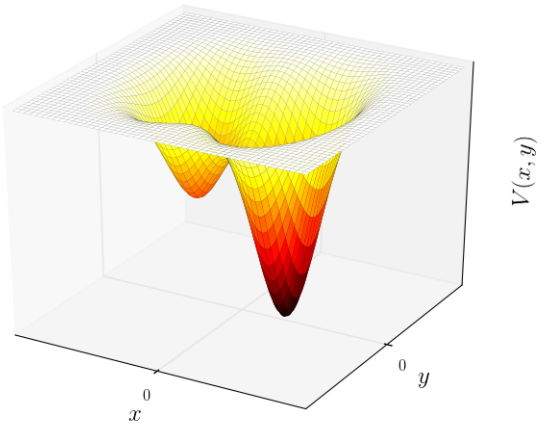



Figure 13.2: A confining potential for a double quantum dot for the $x$ and $y$ coordinates.

scheme is to change the shape of the confining wells and the barrier between the quantum dots, and measure the reaction. The idea was to perform experiments where the depths of the wells are changed over time, but a lack information of parameterizations, time-scales, rapidity of changes, made it a future project.

# Chapter 14

# Results and Analysis

In this chapter the numerical results of the systems introduced in Chapter 13 are presented, analyzed and discussed. To examine the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF)-method we will begin with the single quantum dot in one and two-dimensions. We start by studying the spatial discretization, convergence properties, and analyze the number of orbitals needed for convergence. This is done for the one-dimensional system since the computational costs are much lower, thus enabling us to perform more test. Typical quantities of interest are ground state energies, electron-distributions and simple time-development. All the results are generated using the MCTDHF method unless otherwise stated. For the two-dimensional quantum dots the ground state energy computed with MCTDHF is compared with results calculated using Variational Monte-Carlo (VMC) and Diffusion Monte-Carlo (DMC)[1] methods. Note that this comparison is not exact as the potentials are slightly different due to the use of a shielding parameter in the Coulomb interaction, but it shows the main trends. After the initial test of the single quantum dot, we will investigate the more complex double quantum dot system. Note that some of the results are marked with a asterisk, *. These results have not converged adequately to the convergence criterion used.

All the computations are performed using a single-particle basis of orbitals discretized on a uniform spatial grid. The number of orbitals used as a basis are always chosen to fill the a full shell. Unless otherwise stated the Runge-Kutta-Fehlberg (RK45) integrator is used both for imaginary time propagation and real time propagation.

Most computations were performed on the Abel Cluster, owned by the University of Oslo and the Norwegian metacenter for High Performance Com-

[1]The code-developed for VMC and DMC can be found at `https://github.com/sigvebs/VMC2`.

puting (NOTUR), and operated by the Research Computing Services group at USIT, the University of Oslo IT-department. `http://www.hpc.uio.no/`

## 14.1 The Single Quantum Dot

To model the single quantum dot we use the harmonic oscillator potential, described by Eq. (13.7). The computations are performed in both one- and two-dimensions. The one-dimensional system has the benefit of a much lower numerical cost, and it is therefore easier to explore how the MCTDHF behaves. Using different oscillator frequencies, we can explore both tightly and weakly bound systems, and measure the effect of the correlations on the ground state energy, one-body density of the system, and how MCTDHF manages for at different levels of correlations in the system. The oscillator frequencies we will use are $\omega = 1.0$ for the tightly bound, and $\omega = 0.01$ for the weakly bound system. The interaction between the electrons is modeled using the shielded Coulomb interaction, described in Section 13.2.1, with an interaction strength $\lambda = 1.0$ and a shielding parameter $a = 0.01$. For both the one- and two-dimensional computations the derivatives are calculated using Fourier transformations.

### 14.1.1 The One-Dimensional Quantum Dot

The one-dimensional system allows us more flexibility for exploration compared to the two-dimensional system, due to the grid being quadratic in two-dimensions - making the computations more demanding. We will examine the behavior of the MCTDHF-method by testing a variety of spatial grids and orbitals. Ground state propagations are performed using up to 10 orbitals. For $\omega = 1.0$ the domain of the spatial grid is $\Gamma = [-5, 5]$, and for $\omega = 0.01$ we will use $\Gamma = [-50, 50]$. The initial orbitals are the discretized solution to the non-interacting harmonic oscillator problem, solved on $\Gamma$.

#### Imaginary Time Propagation

Figure 14.1 shows the results of the imaginary time propagation for a variety of resolutions of the grid, orbitals and oscillator frequencies. Figure 14.2 shows the convergence of the energy as a function of the number of imaginary time steps taken the configurations using four and six orbitals. There is a distinct difference between the convergence shown in Figure 14.2a and Figure 14.2b. Using six orbitals the convergence is fast, and unproblematic, but for the four orbital computations there are plateaus of apparent convergence,

the convergence time increases with a finer grid resolution, and for 256 grid points and four orbitals the computation does not converge at all. The difference in energy between 100 steps is less than $10^{-11}$, indicating a stiffness in the problem. This stiffness is more apparent in the $\omega = 0.01$ computations. For some reason the computations using four orbitals are especially affected. As a test the initial orbital basis was exchanged with a randomly chosen basis, but the plateaus of apparent convergence are still present.

Going back to the ground state energies presented in Table 14.1, we see that the weakly bound system converges to a high precision using just 64 grid points. A higher resolution do not seem necessary to describe to problem. This is not the case for the strongly bound system, where the results do not seem to converge within the chosen model space of the grid. For both systems we observe that the results has yet to converge as a function of orbitals, and we could probably get better results by increasing the number of orbitals. The natural orbitals tells us the importance of specific orbitals in the description of the wavefunction. If the natural populations are constant when increasing the number of configurations, the system has converged. Looking at Table 14.2 there is an indication that a minimum of six orbitals are needed to describe the tightly bound system. Increasing the number of orbitals above results in a finer adjustment of the weights.

The electron density is plotted in Figure 14.3. The densities for the weakly bound system converge well within all the grid resolutions. However, the results using two orbitals (one Slater determinant) is startling. Using one Slater determinant causes the electrons to spread out over a large area, and does not describe the system in a good way. Such a two orbital computation is the same as a Hartree-Fock calculation. Besides this case, the electron densities shows a well convergent behavior.

When performing MCTDHF calculations using imaginary time propagation, we are basically finding the optimal set of single-particle orbitals and Slater determinants need to describe the wavefunction. This means that we can now find the minimum number of Slater determinants needed to describe a problem. Figure 14.1 shows the optimal set of Slater determinants for both systems. The strongly bound system appears to converge towards one dominant determinant and three semi-dominant. The weakly bound system tends towards five determinants, but as the number of configurations is increased there are also more low weighted determinants influencing the system, indicating a higher degree of correlation.

| Orbitals | $\omega$ | $E_0[Ha]$ | $K$ | $\omega$ | $E_0[Ha]$ | $K$ | $N_{grid}$ |
|---|---|---|---|---|---|---|---|
| 2 | 1.0 | 6.370523 | 1.000000 | 0.01 | – | – | 64 |
| 4 | 1.0 | 2.742592 | 2.000000 | 0.01 | 0.0699575 | 2.000000 | 64 |
| 6 | 1.0 | 2.757946 | 1.840812 | 0.01 | 0.0695125 | 2.000479 | 64 |
| 8 | 1.0 | 2.739221 | 1.785771 | 0.01 | 0.0692433 | 2.069112 | 64 |
| 10 | 1.0 | 2.725294 | 1.749593 | 0.01 | 0.0692056 | 2.084288 | 64 |
| 2 | 1.0 | 5.084256 | 1.000000 | 0.01 | – | – | 128 |
| 4 | 1.0 | 2.743338 | 2.000000 | 0.01 | 0.0699590 | 2.000000 | 128 |
| 6 | 1.0 | 2.729194 | 1.760420 | 0.01 | 0.0695116 | 2.000888 | 128 |
| 8 | 1.0 | 2.712360 | 1.712892 | 0.01 | 0.0692432 | 2.069164 | 128 |
| 10 | 1.0 | 2.700519 | 1.682707 | 0.01 | 0.0692055 | 2.084238 | 128 |
| 2 | 1.0 | 4.517783 | 1.000000 | 0.01 | – | – | 256 |
| 4 | 1.0 | 2.876705 | 1.806759 | 0.01 | 0.0699590 | 2.000000 | 256 |
| 6 | 1.0 | 2.706676 | 1.700391 | 0.01 | 0.0695100 | 2.001558 | 256 |
| 8 | 1.0 | 2.691345 | 1.658681 | 0.01 | 0.0692430 | 2.069301 | 256 |
| 10 | 1.0 | 2.680994 | 1.632884 | 0.01 | 0.0692055 | 2.084239 | 256 |

Table 14.1: The results of an imaginary time propagation of the one-dimensional quantum dot. The ground state energy is denoted $E_0$ and is measured in Hartrees$[Ha]$, $K$ is the degree of correlation, $\omega$ is the strength of the binding potential and $N_{grid}$ is the number of grid points used.

| | 2 orbitals | 4 orbitals | 6 orbitals | 8 orbitals | 10 orbitals | 12 orbitals |
|---|---|---|---|---|---|---|
| $\phi_0$ | 1.000000 | 0.500000 | 0.709931 | 0.725479 | 0.735772 | 0.740819 |
| $\phi_1$ | – | 0.500000 | 0.249855 | 0.237061 | 0.227409 | 0.223079 |
| $\phi_2$ | – | – | 0.040214 | 0.035885 | 0.034680 | 0.033718 |
| $\phi_3$ | – | – | – | 0.001575 | 0.001420 | 0.001582 |
| $\phi_4$ | – | – | – | – | 0.000719 | 0.000639 |
| $\phi_5$ | – | – | – | – | – | 0.000163 |

Table 14.2: Natural populations for the spin reduced orbitals for the one-dimensional quantum dot with $\omega = 1.0$ and $N_{grid} = 128$. The numbers are the natural populations of the natural orbitals. The left axis indicates the natural orbitals (spin independent), and the top axis indicates the total number of orbitals used in the computations.

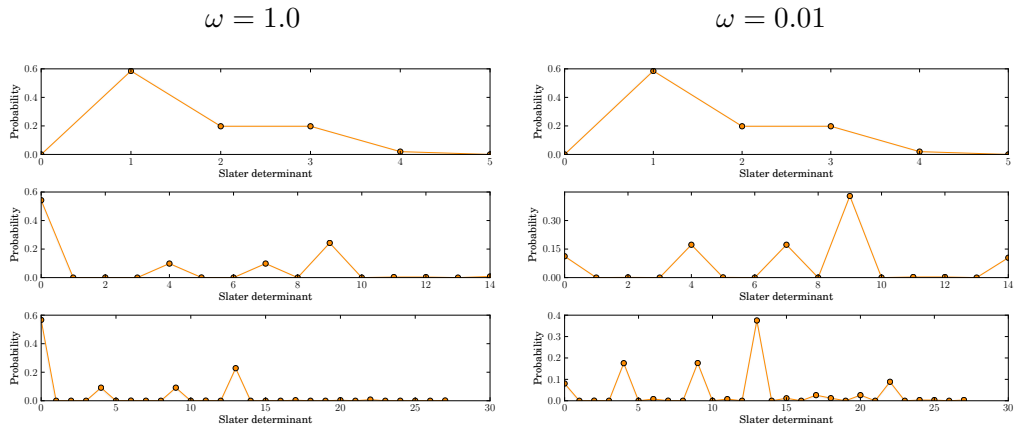



Figure 14.1: The plots show the probability weight of the expansion coefficients (Slater determinants) of the full wavefunction for the one-dimensional quantum dot with $N_{grid} = 128$ for an increasing number of configurations. The results on the left are for the tightly bound system with $\omega = 1.0$, while on the right the we see the results for the weakly bound system with $\omega = 0.01$.

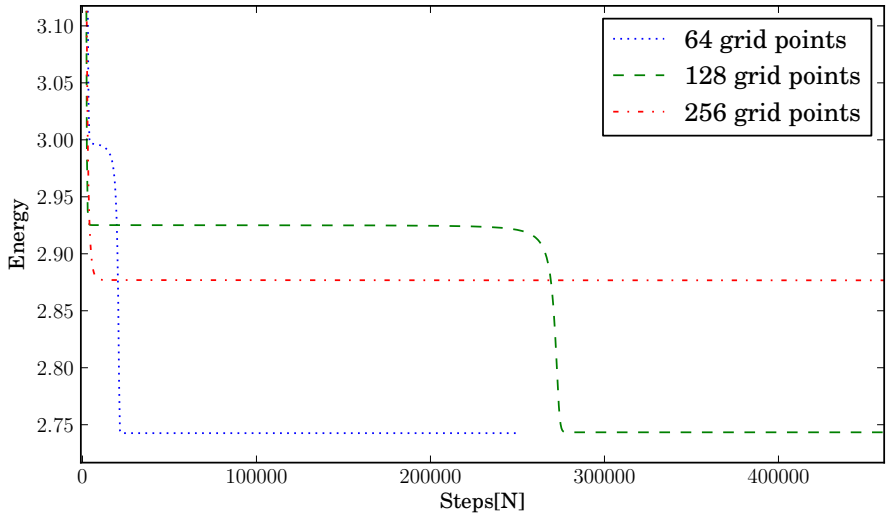
### Time Propagation

We will not perform any time-dependent analysis of the one-dimensional quantum dot as it was studied during the validation of the code by means of comparing the results with published results by Zanghellini [54], see Chapter 12 for details.

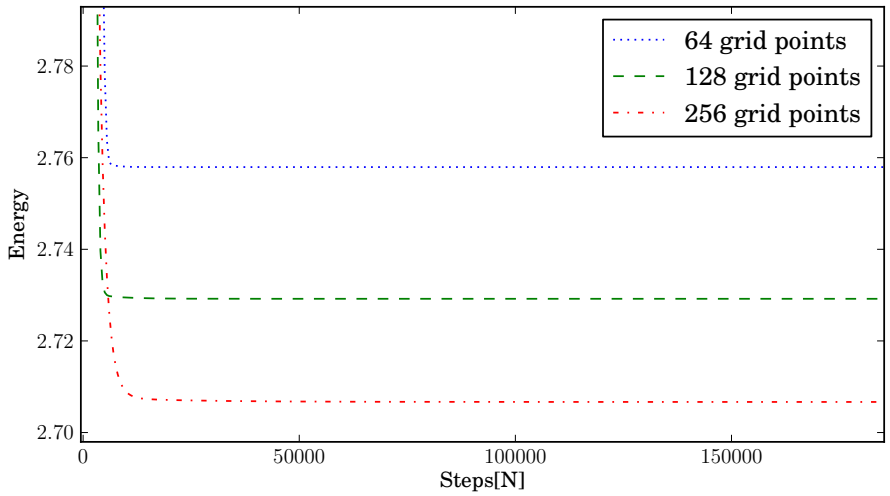
## 14.1.2 The Two-Dimensional Quantum Dot

For the two-dimensional quantum dot computations are performed using the oscillator frequencies $\omega = 1.0$ and $\omega = 0.01$ , on a uniform grid with a resolution of 128 points in both dimensions, totaling 16384 points. Ground state propagations are performed for up to 20 orbitals, using closed shells. For $\omega = 1.0$ the domain of the spatial grid is $\Gamma = [-5, 5] \times [-5, 5]$, and for $\omega = 0.01$ we will use $\Gamma = [-40, 40] \times [-40, 40]$. The initial orbitals are the discretized solutions to the non-interacting harmonic oscillator problem, solved on $\Gamma$.

As an experiment to test the time-development the system is radiated by a laser. Two different frequencies are tested: the first arbitrarily chosen, the second based on the energy gap between the ground state and the first excited state.



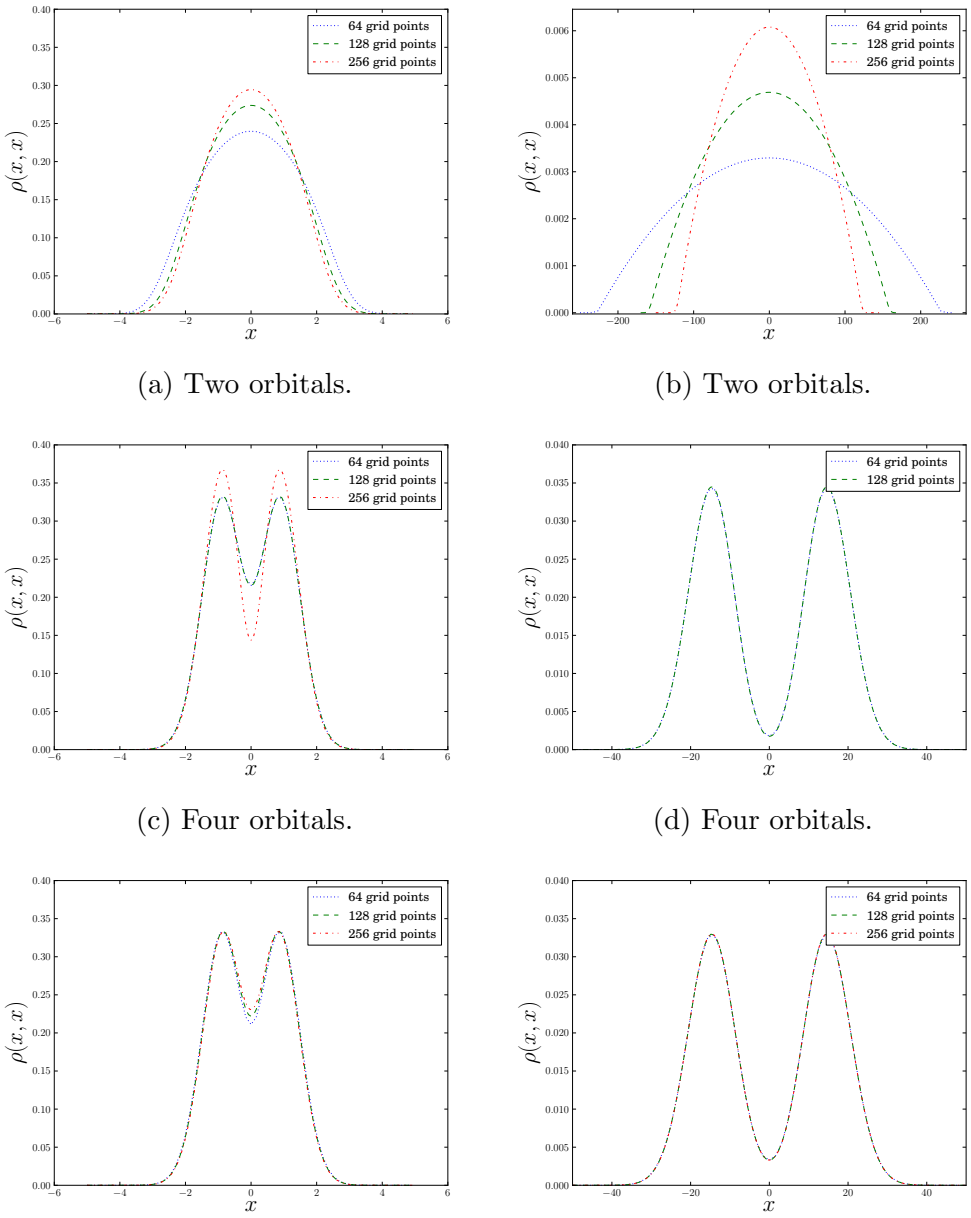

(a) Four orbitals.




(b) Six orbitals.

Figure 14.2: Convergence of the energy for the one-dimensional, two-electron quantum dot with $\omega = 1.0$, for different resolutions of the spatial grid. The computations are performed using imaginary time propagation.

$\omega = 1.00$ $\omega = 0.01$




(a) Two orbitals.

(b) Two orbitals.

(c) Four orbitals.

(d) Four orbitals.

(e) Six orbitals.

(f) Six orbitals.

Figure 14.3: The electron density of the one-dimensional, two-electron quantum dot using different resolutions of the spatial grid. The results in the left column are for the strongly confined system with $\omega = 1.0$. The results shown in column on the right are for the weakly confined systems with $\omega = 0.01$. The densities where computed from the wavefunction found using imaginary time propagation.

### Imaginary Time Propagation

We will start with the results of the strongly confined system, i.e. the system with $\omega = 1$. The imaginary time propagation results are shown in Table 14.3. The convergence of the energy, as function of integration steps, is shown in Figure 14.5. The figure clearly dictate that one determinant is dominating the solution. This is in stark contrast to the one-dimensional results where the weights are spread of several determinants. This is also shown in the degree of correlation. The two-dimensional systems shows less correlations than the one-dimensional. The electron density is plotted in Figure 14.4. For all the configurations the electron density has one distinct peak in the center of the potential. Increasing the number of orbitals does not change the shape notably. A comparison with the one-dimensional electron density, shown in Figure 14.3, demonstrates a qualitative difference between the one and two-dimensional systems - the one-dimensional quantum dot has two peaks.

The natural populations of orbitals are show in Table 14.5. It is clear that one spatial orbital is dominating the solution, but in order to get a fine adjustments to the energy, a multiple of excited orbitals are required. The same trend is seen by observing the dominant number of Slater determinants, shown in Figure 14.5.

The results of the ground state calculations for the weakly bound system is shown in Table 14.3. For $\omega = 0.01$ the convergence, as a function of orbitals, is slower compared to the strongly bound system. This is not surprising as the system shows a higher degree of correlation. Figure 14.6 shows that an additional number of Slater determinants are needed to describe the system. There is no longer one dominant determinant. This is also seen from the ground state energy, where the result using one determinant is far-removed from the multi-determinant computations. The natural populations, shown in Table 14.6, seems to support this interpretation. There is still one dominant spatial orbital, but it is not as dominant as in the strongly bound system. The weights are in general more spread out - this systems needs a higher number of orbitals for an accurate description. The electron densities for some configurations are shown in Figure 14.4. Observing the densities, it is clear that the one determinant solution does not reproduce the electron structure at all. The configurations using more than two orbitals show a circular structure with a hole in the center. The results using six and 12 orbitals show a similar structure; increasing the number of orbitals appears to widen the quantum dot slightly.

Comparing the ground state energies with results generated using VMC and DMC, presented in Table 14.4, we notice that the MCTDHF results are

| Orbitals | $\omega$ | $E_0[Ha]$ | $K$ | $E_1[Ha]$ | $N_{grid}$ |
|---|---|---|---|---|---|
| 2 | 1.0 | 3.197010 | 1.0 | – | 16384 |
| 6 | 1.0 | 3.054969 | 1.117140 | 3.7699 | 16384 |
| 12 | 1.0 | 3.026028 | 1.110113 | 3.7682 | 16384 |
| 20* | 1.0 | 3.018412 | 1.106866 | – | 16384 |
| 2 | 0.01 | 0.115818 | 1.0 | – | 16384 |
| 6 | 0.01 | 0.075030 | 2.022636 | 0.0798381 | 16384 |
| 12* | 0.01 | 0.073920 | 2.273976 | 0.0760413 | 16384 |

Table 14.3: Imaginary time propagation results for a two-dimensional quantum dot, were $E_0$ is the ground state energy and $E_1$ indicates the energy of the first excited state, both given in units of Hartrees[Ha]. The degree of correlation is given by $K$ and the confinement strength is given by $\omega$.

| Method | $\omega$ | $E_0[Ha]$ |
|---|---|---|
| $VMC$ | 0.01 | 0.07407(3) |
| $DMC$ | 0.01 | 0.073818(7) |
| $VMC$ | 1.0 | 3.00038(4) |
| $DMC$ | 1.0 | 3.00001(2) |

Table 14.4: Ground state energies for the quantum dot in two-dimensions computed using Variational Monte Carlo (VMC) and Diffusion Monte Carlo (DMC).

somewhat higher. Note that the results computed using VMC and DMC did not use a shielded Coulomb interaction - a direct comparison is not entirely correct. One point of notice, though, is that the weaker bound system produces results better than VMC, and quite close to the DMC results. DMC is known to be one of the most accurate and efficient methods for finding the ground state energies.

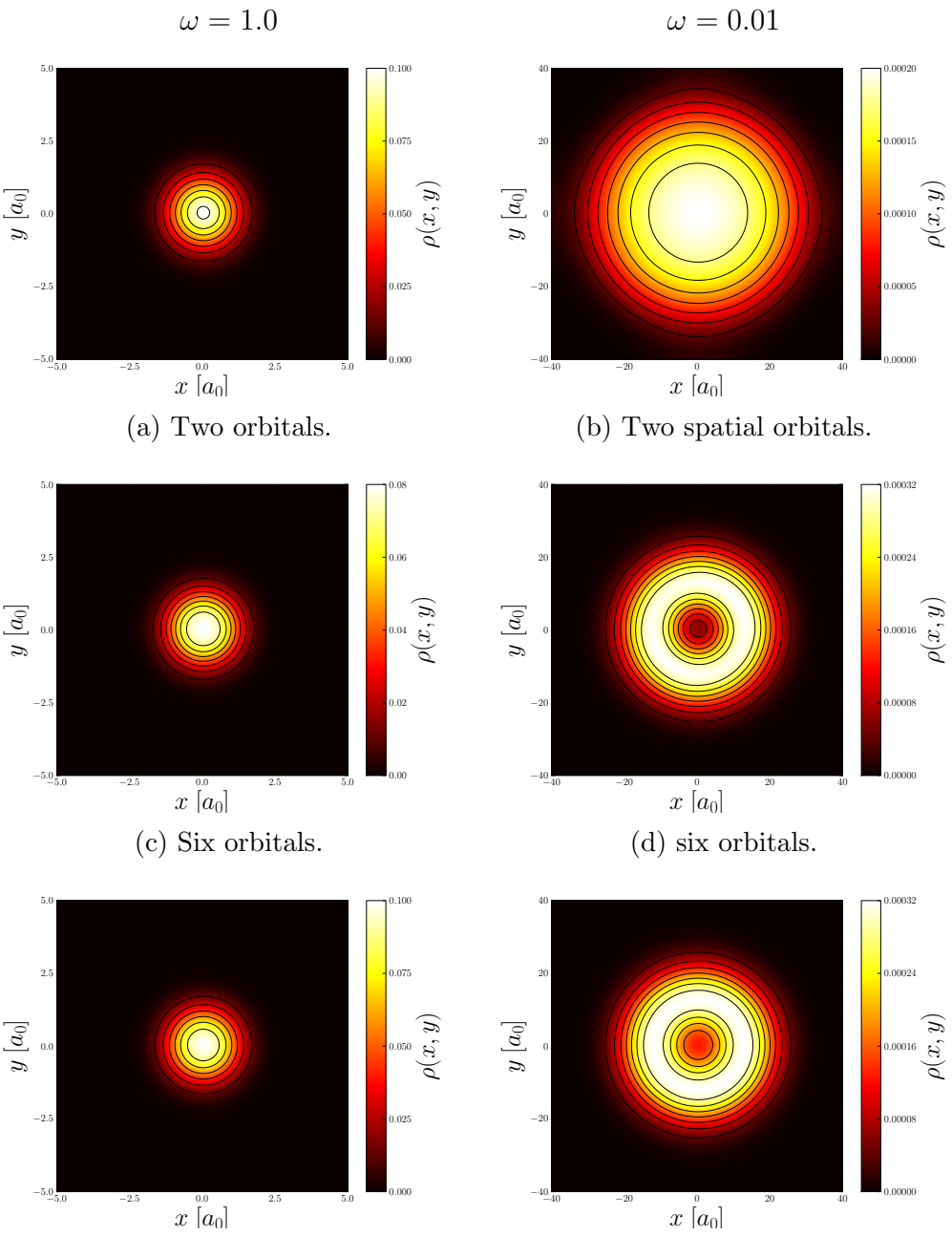
When comparing the strongly and weakly bound systems for the number of steps needed for convergence, we observe, in general, that the weakly bound systems shows a higher degree of stiffness and needs more time to converge. In most cases the results appear to converge correctly, but some of the results have not shown an adequate degree of convergence, and could be only partially converged. For the two-dimensional computations there are no trace of the plateaus of apparent convergence, just a slowly descending energy after the initial convergence. This descent can be very long, though.

| | 2 orbitals | 6 orbitals | 12 orbitals | 20 orbitals* |
|---|---|---|---|---|
| $\phi_0$ | 1.000000 | 0.945330 | 0.948542 | 0.950175 |
| $\phi_1$ | – | 0.027335 | 0.023009 | 0.022256 |
| $\phi_2$ | – | 0.027335 | 0.023009 | 0.021904 |
| $\phi_3$ | – | – | 0.004212 | 0.003866 |
| $\phi_4$ | – | – | 0.000614 | 0.000581 |
| $\phi_5$ | – | – | 0.000614 | 0.000574 |
| $\phi_6$ | – | – | – | 0.000247 |
| $\phi_7$ | – | – | – | 0.000236 |
| $\phi_8$ | – | – | – | 0.000092 |
| $\phi_9$ | – | – | – | 0.000070 |

Table 14.5: Natural populations for the spin reduced orbitals for the two-dimensional, two-electron quantum dot with a confining strength $\omega = 1.0$. The numbers are the natural populations of the natural orbitals. The left axis indicates the natural orbitals (spin independent), and the top axis indicates the total number of orbitals used in the computations.

| | 2 orbitals | 6 orbitals | 12 orbitals |
|---|---|---|---|
| $\phi_0$ | 1.000000 | 0.661006 | 0.610419 |
| $\phi_1$ | – | 0.172232 | 0.182973 |
| $\phi_2$ | – | 0.166762 | 0.182942 |
| $\phi_3$ | – | – | 0.010258 |
| $\phi_4$ | – | – | 0.006707 |
| $\phi_5$ | – | – | 0.006702 |

Table 14.6: Natural populations of the spin reduced orbitals for the two-dimensional, two-electron quantum dot with a confining strength $\omega = 0.01$.




(a) Two orbitals.

(b) Two spatial orbitals.

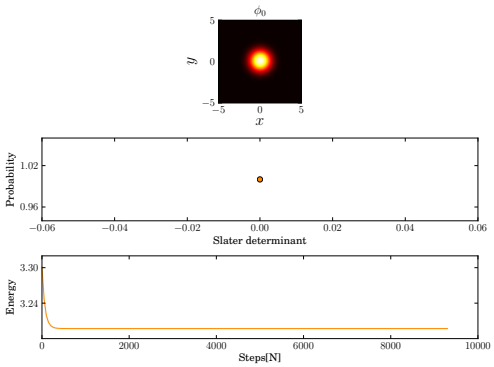
(c) Six orbitals.

(d) six orbitals.

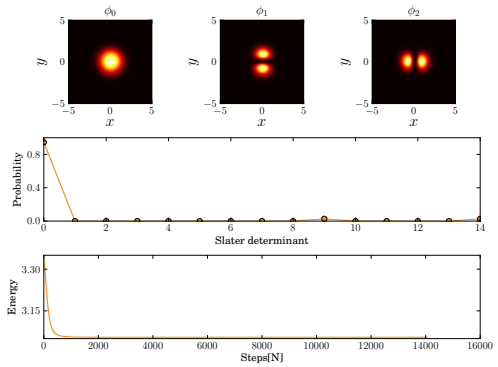
(e) Twelve

(f) Twelve orbitals.

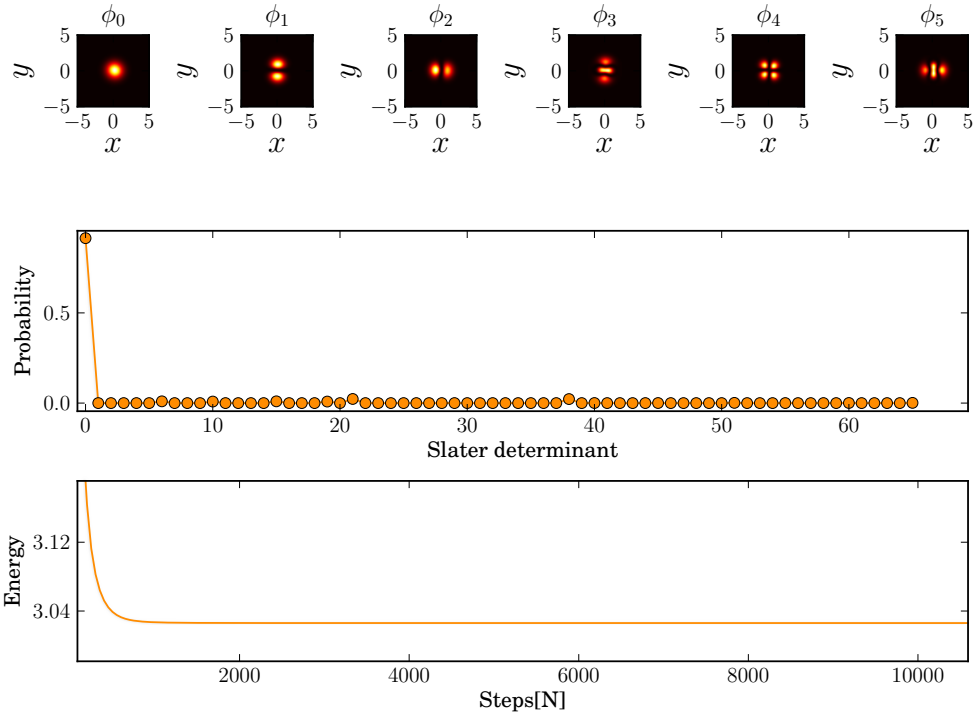
Figure 14.4: Electron densities of the two-electron quantum dot. The results on the left are for the strongly confined system with $\omega = 1.0$. On the right we have the results for the weakly confined system with $\omega = 0.01$. The densities where computed from the wavefunction found using imaginary time propagation.




(a) 2 orbitals.

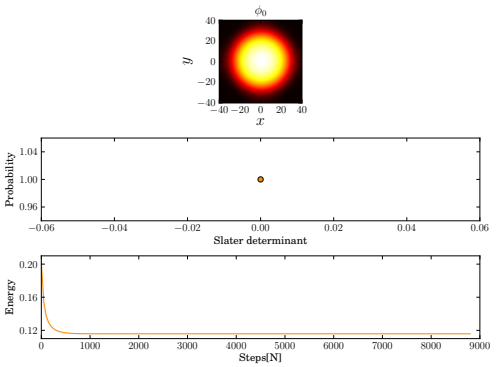



(b) 6 orbitals.

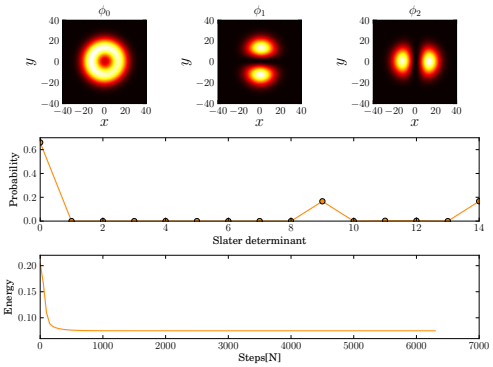



(c) 12 orbitals.

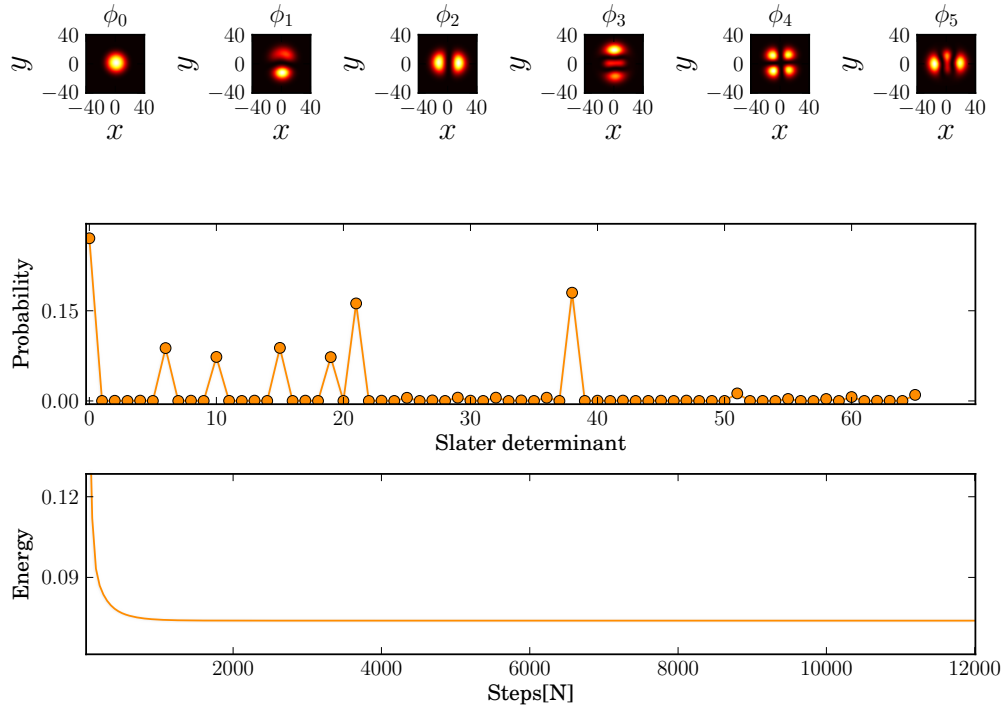
Figure 14.5: Energy convergence of the imaginary time propagation for the two-dimensional single quantum dot with $\omega = 1.0$. The spatial orbitals and the expansion coefficients (Slater determinants) are shown at the last time step.




(a) 2 orbitals.




(b) 6 orbitals.




(c) 12 orbitals.

Figure 14.6: Energy convergence of the imaginary time propagation for the two-dimensional single quantum dot with $\omega = 0.01$. The spatial orbitals and the expansion coefficients (Slater determinants) of the full wavefunction are shown at the last time step.

### Time Propagation

As an experiment in time evolution the quantum dot is exposed to a short laser pulse. The laser is linearly polarized along the $x$-axis, and is modeled by Eq. (13.8). The amplitude is set to $E_0 = 1.0$ . Two difference frequencies are tested: the first arbitrarily chosen, the second is based on the energy gap between the ground state and the first excited state. For the arbitrarily chosen frequency we are not expecting the system to excite, it should only perturbate the system. For the second test, the frequency, which we will call the excitation frequency, is decided by the energy gap between the energy-states of the quantum dot. If set correctly, the quantum dot should absorb the energy of the laser, thereby exciting the system in a stepwise manner. Basically, we are modeling a resonance phenomena. We are expecting the quantum dot to oscillate at a higher frequency and amplitude when radiated by the excitation frequency compared to the arbitrary frequency.

For the arbitrary frequency the angular frequency is set to $\Omega = 8.0$. The excitation frequency is calculated using the difference in energy between the ground state and the first excited state given in Table 14.3. For the two and six orbital computations the angular excitation frequency is set to $\Omega = 0.7149$, and for the computations using twelve orbitals we use $\Omega = 0.7422$.

The results of the radiation, using the arbitrary frequency, are shown in Figure 14.7 and Figure 14.8. As expected the laser only perturbs the system about the ground state. The shape of the results in Figure 14.7 are the same for all the configurations. The difference is only the amplitude of the oscillations.

Figure 14.9 shows the overlap of the time-evolved state and the initial state of the quantum dot, radiated by the laser using the excitation frequency. The system is rapidly thrown out of the initial state, containing no remnants of the ground state. This is exactly the result expected when radiating a systems with a laser using an energy shift corresponding to the excitation energy of the system. The absorption of energy is seen in Figure 14.10, where we see a step-like excitation of energy. Figure 14.11 shows the time evolution frame-by frame. The quantum dot absorbs the energy, and the effect of the absorption is that the whole quantum dot starts oscillating about the center of the potential. The time evolution is computed correctly, but the size of the grid is actually to small, which introduces some unphysical boundary effects when the quantum dot gets close to the boundaries. This is not a problem, however, as this only happens quite late in the simulation. The shape of the quantum dot should be conserved, but the last frames show a distortion of the quantum dot when it gets to close to the boundary.

Comparing Figure 14.12, showing time evolution using two orbitals, with

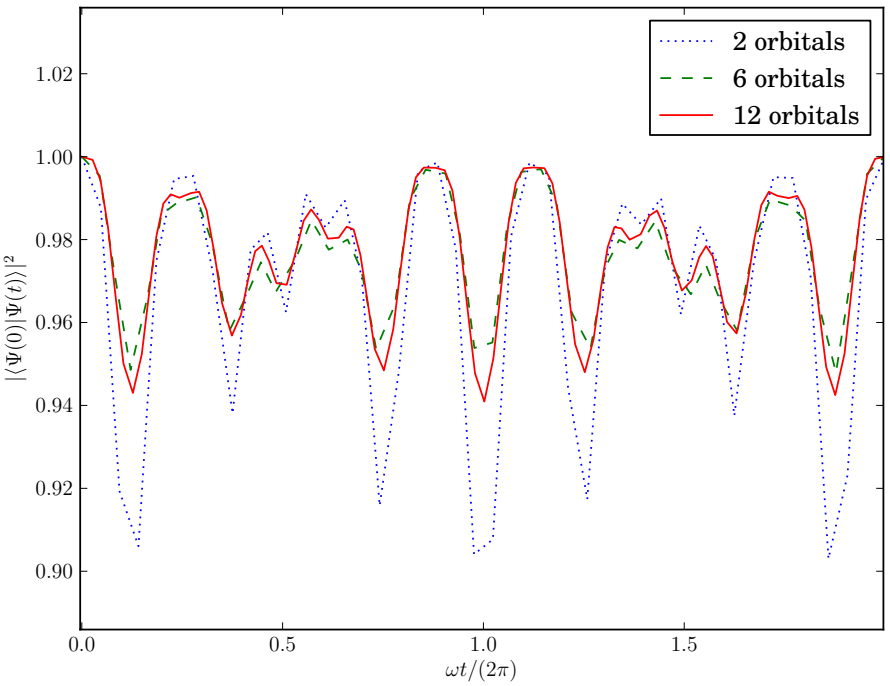



Figure 14.7: Overlap with the initial state for a the two-dimensional, two-electron quantum dot radiated by a laser with an arbitrarily chosen angular frequency $\Omega = 8.0$.

Figure 14.11, where twelve orbitals are used, there does not seem to be any large qualitative differences between the results of the time evolution.

Finally, Figure 14.13 shows an example of the step length chosen by the Runge-Kutta-Fehlberg solver to keep the error in the propagation below a threshold of $10^{-12}$.

## 14.2 The Double Quantum Dot

For our simulations of the double quantum dot we will utilize different potentials for the one- and two-dimensional computations. The interaction between the electrons is, as before, modeled using the shielded Coulomb interaction, described in Section 13.2.1. The interaction strength, $\lambda$, and the shielding parameter, $a$, will be defined by the material used in the simulated. For both the one- and two-dimensional computations the derivatives are calculated using Fourier transformations.

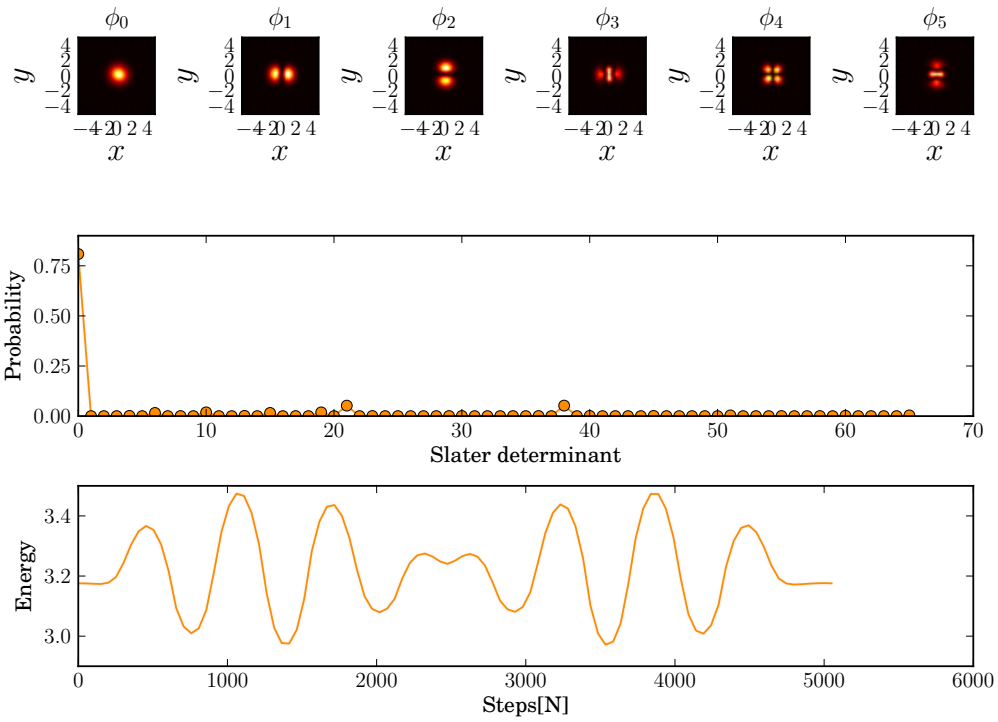



Figure 14.8: The energy as a function of time for the two-dimensional, two-electron quantum dot using a one-particle basis containing 12 orbitals. The system is radiated by a laser with an arbitrary chosen frequency. The spatial orbitals and the expansion coefficients (Slater determinants) of the full wavefunction are plottet for the last time step.

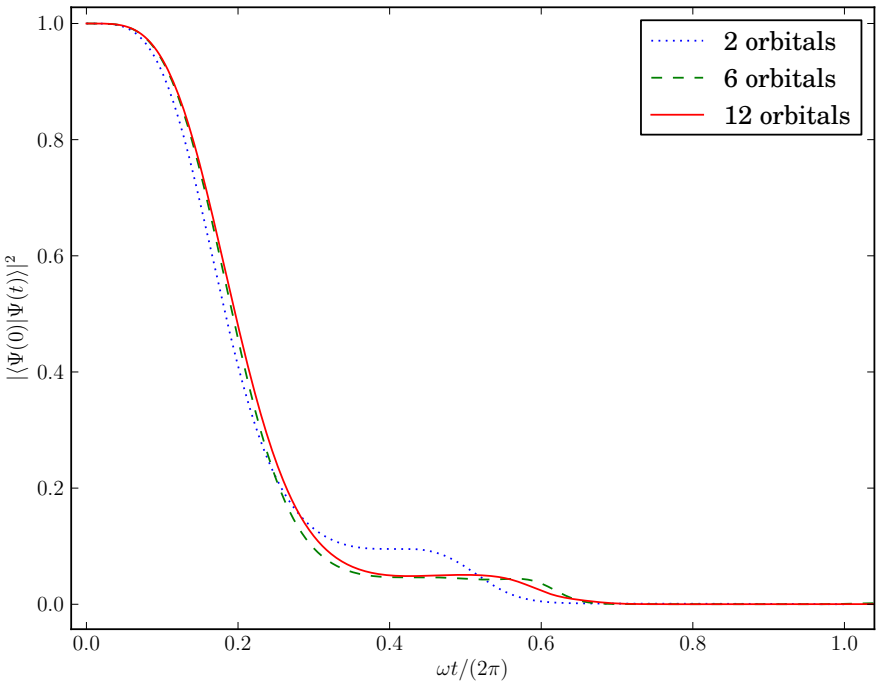



Figure 14.9: The two-dimensional, two-electron quantum dot is radiated by a laser using a frequency based on the difference in energy between the ground state and the first excited state of the undisturbed system. The figure shows the overlap with the initial state.

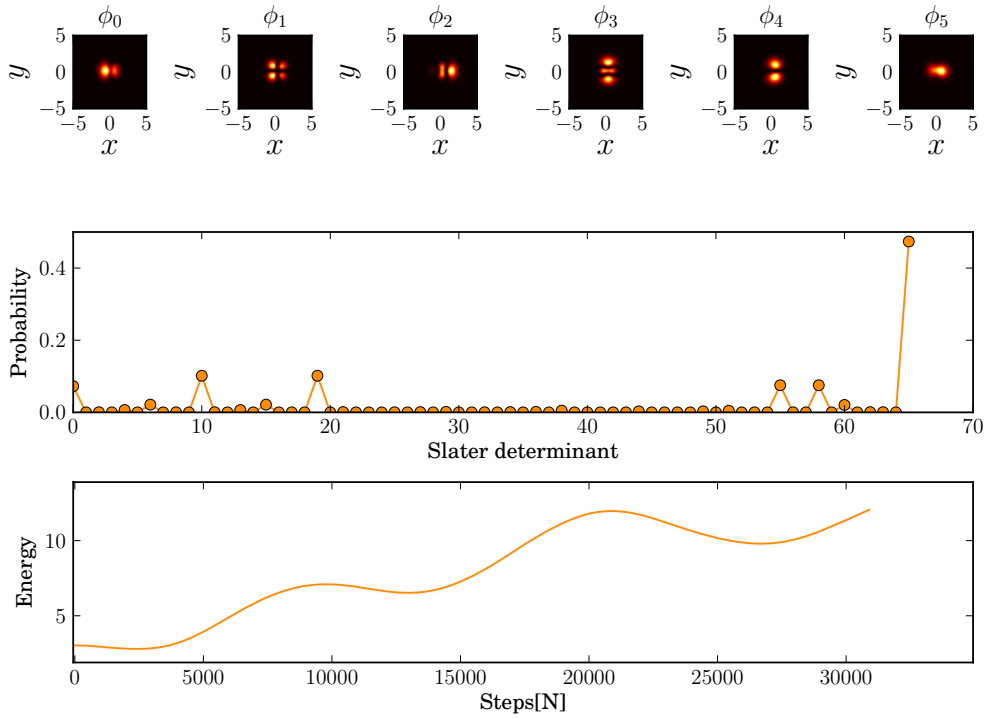



Figure 14.10: The energy as a function of time for the 12 orbital system radiated by a laser using a frequency tuned to the system. The spatial orbitals and the expansion coefficients (Slater determinants) of the full wavefunction are plotted for the last time step.

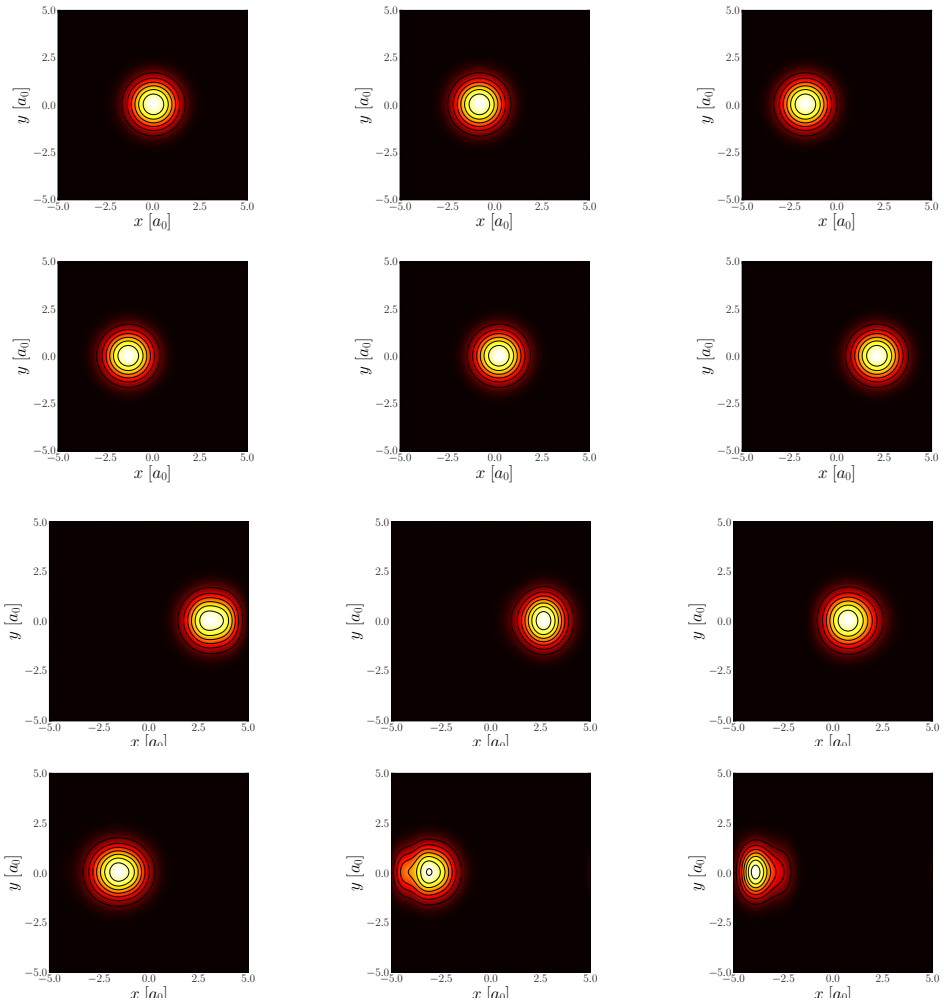



Figure 14.11: Time evolution of the two-dimensional, two-electron quantum dot, described in a one-particle basis containing 12 orbitals, radiated by a short laser pulse using the excitation frequency. The time evolution is read from left to right, with the top left showing the initial state, and the bottom right shows the result for the last time-step. Between each frame 250 time-iterations are performed.

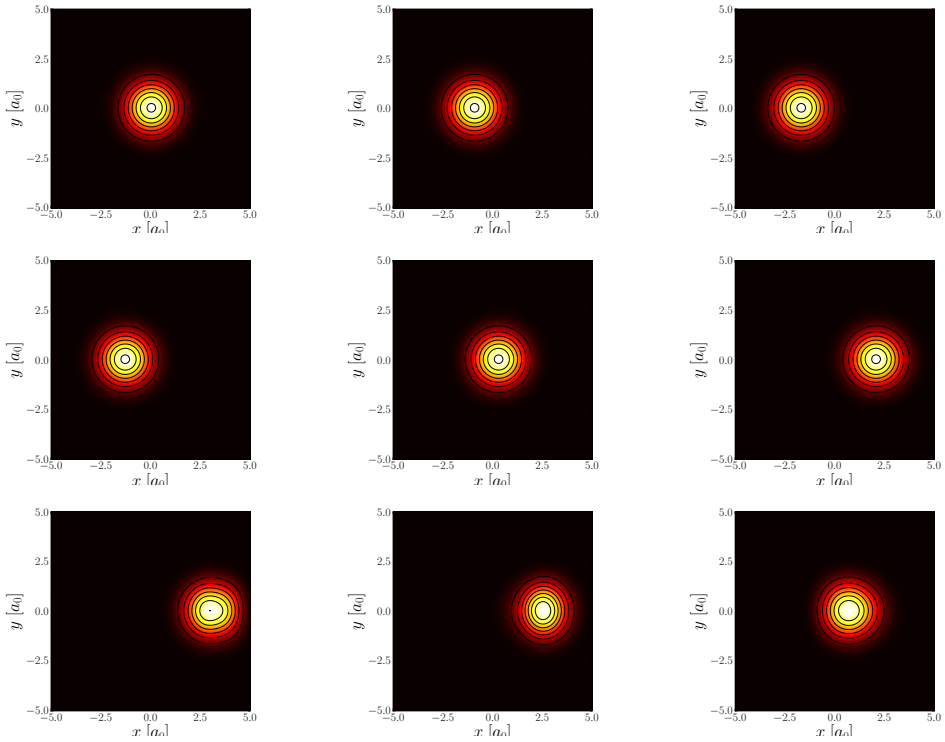
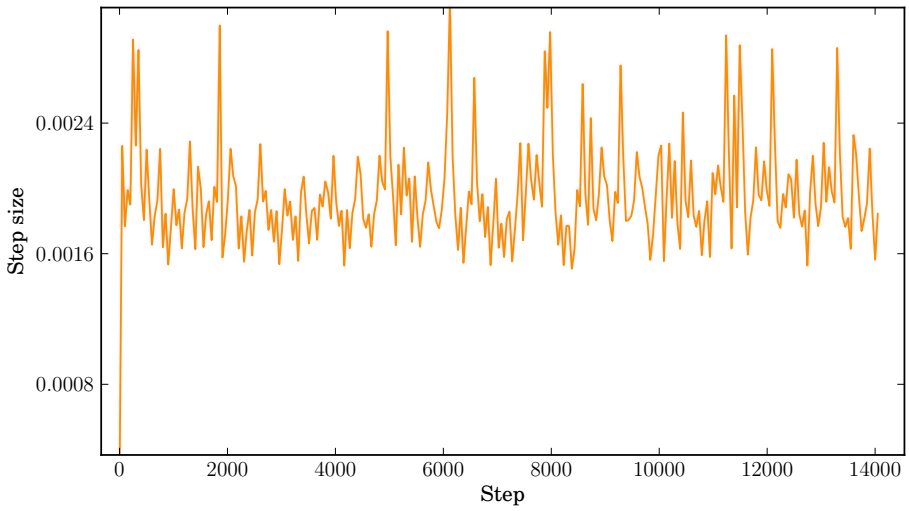
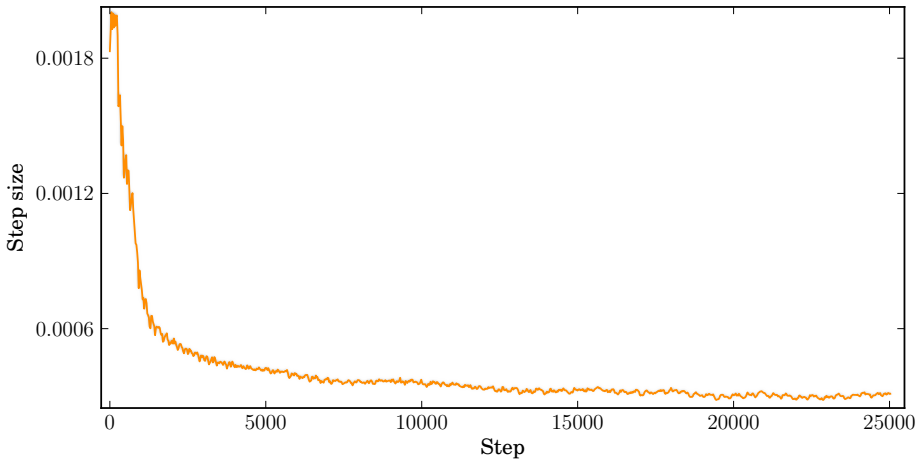



Figure 14.12: Time evolution of the two-dimensional quantum dot, described in a one-particle basis containing two orbitals, radiated by a short pulse laser using the excitation frequency. The time evolution is read from left to right, with the top left showing the initial state, and the bottom right shows result for the last time-step. Between each frame 250 time-iterations are performed.




(a) The step length used during imaginary time propagation of the two-dimensional quantum dot.




(b) The step length used during time propagation of the two-dimensional quantum dot radiated by a laser with an arbitrary frequency.

Figure 14.13: Step length as a function of the number of steps taken for the two-dimensional quantum dot. The step length chosen by the Runge-Kutta-Fehlberg integration scheme to keep the error below a certain threshold.

### 14.2.1 The One-Dimensional Double Quantum Dot

For the double quantum dot in one-dimension we are using the potential given by Eq. (13.11). The parameters are chosen to match the ones used in Kryvi's master thesis [12]. The distance, $d$, between the wells is set to $d = 8$. For the shielded Coulomb potential we will use $\lambda = 0.6$ and $a = 0.1$. The grid is uniform distributed on the domain $\Gamma = [-15, 15]$.

#### Imaginary Time Propagation

Table 14.7 shows the results of an imaginary time propagation of two electrons trapped in a double well. The convergence to the ground state, as a function of imaginary time steps, is shown in Figure 14.14. Comparing with Kryvi's result for the ground state, $E_0 = 1.0467$, the results coincides well for four orbitals. We observe a rapid convergence with the number of orbitals; the difference in energy between four and six shells are of order $10^{-6}$. Interestingly there is a better correspondence between the results using a higher number of orbitals, independently of the grid-basis used.

Table 14.8 shows the natural population of the orbitals. As with the energy, there is a clear indications that the system is described accurately using only four orbitals as the weight is almost exclusively on the two first spatial orbitals.

The electron density is shown in figure 14.15. We see the same here - the electron density of the two-orbital computations does not describe the system in an adequate way.

Comparing the results for the single quantum dot with the ones for the double quantum dot, we see that the double quantum dot actually converges faster as a function of orbitals, towards a stable ground state energy. This convergence is much slower for the single quantum dot. But the comparison is note entirely fair as the potentials are quite different.

#### Time Propagation

Using the ground state found in the previous section as the initial state, the system is radiated by a laser described by Eq. (13.11). The amplitude of the laser is $E_0 = 1$. The angular frequency is set arbitrarily to $\Omega = 8.0$. The purpose of the radiation is to observe how the system reacts to a disturbance. To illustrate the reaction, Figure 14.16 shows the overlap with the initial state. The figure clearly shows that using one Slater determinant - that is the equivalent of using the time-dependent Hartree Fock method, does not produce acceptable results. By increasing the number of shells by one, the results seems to converge towards the correct solution. This is also seen in

| Orbitals | $E_0[Ha]$ | $K$ | $N_{grid}$ |
|---|---|---|---|
| 2 | 2.112064 | 1.000000 | 32 |
| 4 | 1.046950 | 2.000000 | 32 |
| 6 | 1.046948 | 1.999999 | 32 |
| 2 | 1.810520 | 1.000000 | 64 |
| 4 | 1.046727 | 2.000000 | 64 |
| 6 | 1.046726 | 2.000000 | 64 |
| 2 | 1.703949 | 1.000000 | 128 |
| 4 | 1.046727 | 2.000000 | 128 |
| 6 | 1.046726 | 2.000001 | 128 |

Table 14.7: Results of imaginary time propagation for a one-dimensional quantum dot, were $E_0$ is the ground state energy. The degree of correlation is given by $K$.

| | 2 orbitals | 4 orbitals | 6 orbitals |
|---|---|---|---|
| $\phi_0$ | 1.000000 | 0.500312 | 0.500504 |
| $\phi_1$ | – | 0.499688 | 0.499495 |
| $\phi_2$ | – | – | 0.000001 |

Table 14.8: Natural populations of the spin reduced orbitals for the one-dimensional double quantum dot.

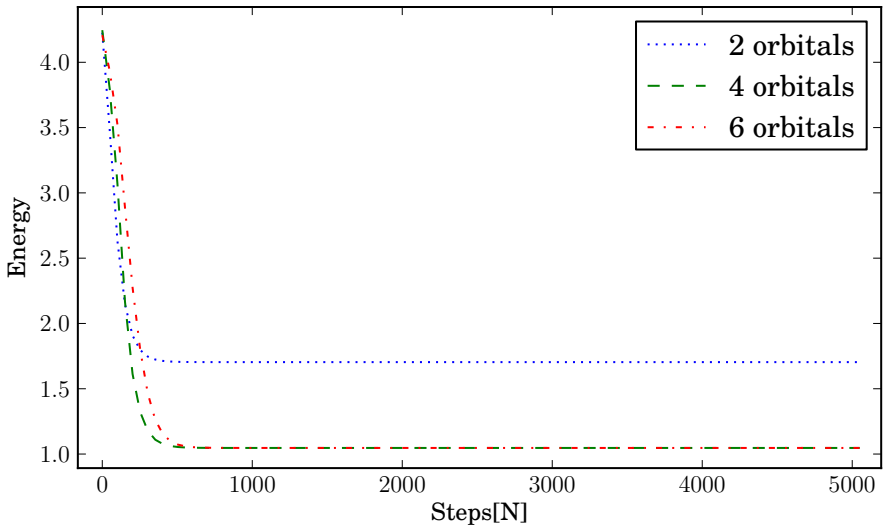



Figure 14.14: Energy convergence for two electrons trapped in a one-dimensional double well potential shown for an increasing number of orbitals/configurations.

the ground state propagation results. But bear in mind that the ground state found using imaginary time propagation is used as a starting point, and for the two-orbital case the ground state wavefunction is very different from the ones generated using more orbitals.

### 14.2.2 The Two-Dimensional Double Quantum Dot

As a final test of the implementation we will simulate a double quantum dot in two-dimensions. The double quantum dot is modeled after the potential given by Eq. (13.12). The idea was to change the depth of the wells over time, and measure of the system reacts, but due to a lack of information how to model time evolution for such a system, we have only performed an imaginary time propagation. All calculations are performed on a uniform grid $\Gamma = [-9.0, 9.0] \times [-4.5, 4.5]$, with a resolution of $128 \times 64$ points, totaling 8192 grid points. The choice of parameters are influenced by reference [57], where they are simulating a double quantum dot for use in quantum computers.

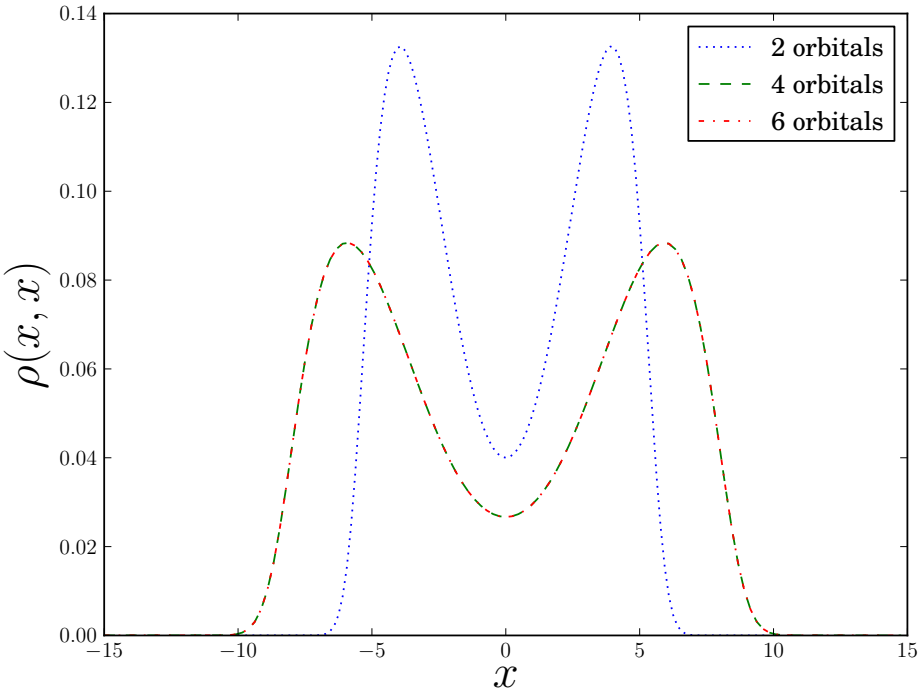



Figure 14.15: Electron density for two electrons trapped in a one-dimensional double well potential shown for an increasing number of orbitals/configurations.

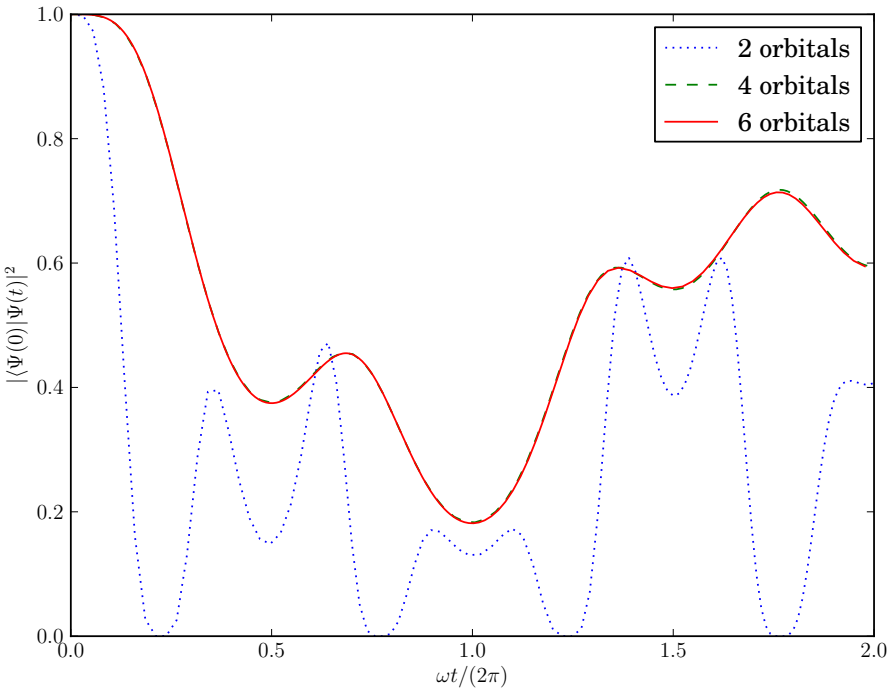



Figure 14.16: Overlap with the initial state and the time propagated state for a two electron system in a one-dimensional double well potentials, shown for an increasing number of orbitals/configurations. The system is radiated by a laser with frequency $\Omega = 8.0$.

| Orbitals | $E$[50meV] | $K$ | $N_{grid}$ |
|---|---|---|---|
| 2 | $-0.97619$ | 1.0 | 8192 |
| 6 | $-0.99150$ | 1.68018 | 8192 |
| 12* | $-0.99150$ | 1.67913 | 8192 |

Table 14.9: The results of an imaginary time propagation of the two-dimensional, double quantum dot. The ground state energy is denoted $E_0$, $K$ is the degree of correlation and $N_{grid}$ is the number of grid points used.

The parameters we used are

$$\begin{array}{ll} V_a = -1.0 & V_b = -1.0 \\ V_c = 0 & l_x = 2.5 \\ l_y = 2.0 & L_{bx} = 2.0 \\ L_{by} = 2.0 & a = 3.4453\ . \end{array}$$

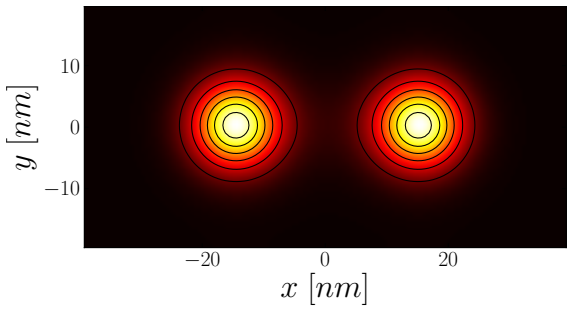
The Coulomb interaction strength is set to $\lambda = 0.42079$ and the shielding parameter is set to $a = 0.01$. All parameters are given in dimensionless units, with an energy scaling $V_0 = 50$meV.

**Imaginary Time Propagation**

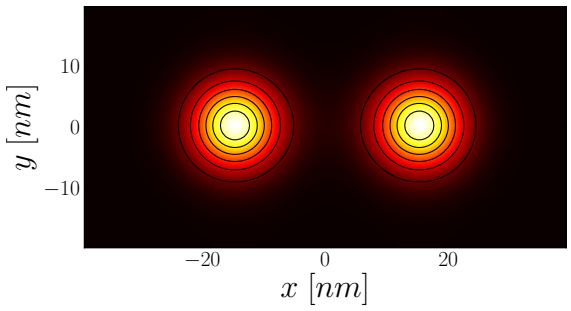
Table 14.7 shows the results of an imaginary time propagation of for the double quantum dot. We see that the energy converges quickly as a function of orbitals, but note that the result for 12 orbitals has not converged to our convergence criterion. It could be that the energy would have fallen lower down if the computation was allowed more time. The results show that the double quantum dot has a stronger degree of correlation than the single quantum dot, by comparing the degree of correlation. The electron densities are shown in Figure 14.17. An interesting point is that the single configuration actually manages to reproduce the electron density quite well, much better than the one-dimensional calculation.

## 14.3 Discussion

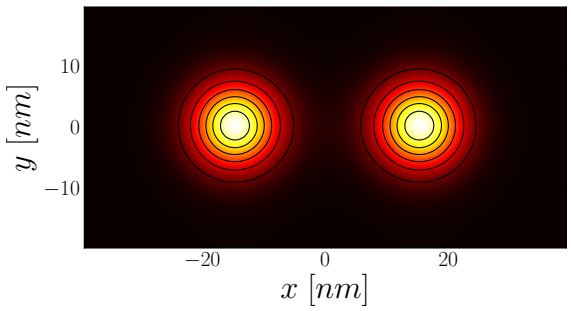
In the previous sections we have presented the results of computations on quantum dots confined in various potentials and simulated in one- and two-dimensions. The experiments were limited in the extent that only two-electron systems were studied. But the results were interesting in themselves, and show us that the MCTDHF method works. We will now discuss some of the results in the following subsection.




(a) Two orbitals.




(b) Six orbitals.




(c) Twelve orbitals.

Figure 14.17: Electron densities of the double quantum dot. The densities where computed from the wavefunction found using imaginary time propagation.

### A Comparison of MCTDHF with other Many-Body Methods

First, the Time-Dependent Hartree-Fock (TDHF) is an example of a method that cannot describe correlations well. In the limit of one configuration, MCTDHF is the same as TDHF. On the other hand, if the number of configurations tends towards infinity, we regain the standard propagation method. By increasing the number of configurations, we should see a better representation of the correlation effects, and in general represent the wavefunction more accurately. Correlations play a stronger role when the potential energy is dominating the problem. This is typical of weakly bound systems, like the ones confined in a harmonic oscillator with $\omega = 0.01$. Checking the degree of correlation, $K$, in Table 14.1, Table 14.3 and Table 14.7, we can conclude that this indeed the case. For the weakly bound single quantum dot and double well systems the degree of correlation is high. For these systems the TDHF method should not produce good results. A quick glance at the electron density plots, i.e., Figure 14.15 and Figure 14.4,shows us that, indeed, TDHF does not produce good results for these systems. The TDHF results cannot describe the structure - one determinant does not produce any meaningful results. However, if we increase the number of configurations in the MCTDHF computations, the electron densities shows more structure and describes the correlation effect better. In most of our computations we see that the configuration level above Hartree-Fock often reproduces most of the correlations, and increasing the configuration space above this only gives finer adjustments. On the other hand, if we look at the results of the strongly confined quantum dots in two-dimensions, the electron distribution and time evolution follows the same shape for the TDHF results as the calculations performed using more configurations. This is seen both in the imaginary time results and the time propagation results. The only difference is the amplitude, as seen in Figure 14.16.

Comparing the ground state energies found for the two-dimensional quantum dot, with results computed using VMC and DMC, we see that the results are in agreement. DMC is considered as one of the most accurate methods for finding the ground state of a system. The results computed in this thesis are close[2] to the DMC results. Nevertheless, DMC is a faster and more accurate method - but it cannot be used in time-dependet studies. Using MCTDHF to compute the ground state is not efficient, if that is the only purpose. We computed ground states to test the MCTDHF method, and to explore convergence behavior and create initial states for use in time propagation. When compared with more accurate methods, like DMC, the MCTDHF-method is

[2] Do keep in mind that the Coulomb interaction used in VMC and DMC does not include a shielding parameter.

slow and produced results with higher energy. It also has problems with singularities due to the finite grid, and a shielding parameter must be introduced in the Coulomb interaction. However, if the energy is not the most important quantity, the MCTDHF-method is *the* method to use for time-dependent study of fermionic quantum systems.

### Imaginary Time Propagation

There are two types of convergence we have to take into account when solving the MCTDHF equations of motion using imaginary time: the convergence of the energy as a function of the configuration space, and the convergence of the iterative solution to the MCTDHF equations within that configuration space. Looking at Figure 14.2, we see examples of both these cases. The figures show convergence of the energy for two different computations, both performed using imaginary time propagation, the only difference is the size of the configuration space. The top figure shows the results of an energy calculation using four orbitals, where the energy is struggling to converge and we are seeing plateaus of apparent convergence. The figure on the bottom shows the same system, computed using six orbitals, and it has no convergence problems. One possible reason is that the initial wavefunction, using six orbitals, has a larger overlap with the ground state, whereas in the four orbital case the overlap is almost non-existent.

We have seen that some of the system are described very well using only a few orbitals, while others need a larger basis to converge. In some cases the Hartree-Fock case produces adequate results, but be careful, if there are correlations in the system it will not produce good results. For our ground state computations the first few orbitals usually contains over 90% of the energy. Increasing the basis gives smaller contributions for every step.

For some of the systems, like the one-dimensional quantum dot and the one-dimensional double quantum dot, we have explored different resolutions of the spatial grid. The effect of using various grid resolutions is very system dependent. The one-dimensional double quantum dot converges rapidly to the solution within a configuration space, while the one-dimensional single quantum dot needs a still higher resolution to represent the solution correctly. There are two obvious sources of error: the choice of differential operators and calculation of the integrals elements. A Fourier transformation is used to compute the second derivative, and it should be close to numerically exact for the grids we have used. The interaction elements are calculated using trapezoidal integration, which has a higher source of error, and is probably causing some of the discrepancies. It is, however, somewhat strange that the double quantum dot converges better as a function of the grid resolution

since the spatial domain is larger.

If we look at the number of configurations needed to describe the one- and two-dimensional quantum dots with $\omega = 1.0$, shown in Figure 14.1 and Figure 14.5, we see that the two-dimensional system is actually better described by one determinant. By checking the degree of correlation for both systems it clear the one-dimensional system is stronger correlated than the two-dimensional system. This means that the Coulomb repulsion in one-dimension has a larger impact on the electrons. A comparison of the electron density plots, shown in Figure 14.3 and Figure 14.4, also shows this effect. In one dimension the electrons are pushed further apart, resulting in two maximum points. In two dimensions the electrons have more options for possible radial distances, creating a smaller quantum dot.

### Time Evolution

The results of the time evolution shows the same trends as with imaginary time propagation. For the strongly correlated systems the Hartree-Fock results are useless, and for the less correlated systems Hartree-Fock seems to produce adequate results. We performed some numerical experiments where the two-dimensional quantum dot was radiated using different frequencies. The results of the time evolution reproduced the physics we expected: using the right frequencies excites a system, choose the wrong frequencies and the system is only slightly disturbed. We did, however, have some problems with the spatial domains. If we look at Figure 14.11 we see some unnatural boundary effects. A better choice of discretization would be to increase the range and resolution along the polarization axis, thereby avoiding these kind of boundary effects.

# Chapter 15

# Summary and Outlook

The aim of this mater's thesis was to solve the time-dependent Schrödinger equation for quantum dots using the Multiconfiguration Time-Dependent Hartree-Fock (MCTDHF)-method. First, we started by reviewing the basic quantum mechanics needed to solve the many-body problem. We then briefly examined the most common method used in time evolution of quantum systems. The method of choice, MCTDHF, was reviewed in detail. Using C++, we implemented the MCTDHF method, using an object-oriented approach. The whole development process is comprehensively documented so that anyone interested in the implementation can benefit from our experience. The code developed is available under a GPL license and can be found at `https://github.com/sigvebs/MCTDHF` for anyone to look at, use or expand. We verified the implementation of the MCTDHF method by comparing results generated using our code with published results by Zanghellini *et al* [54]. The code has successfully produced results for two electron quantum dots, in one and two-dimensions. We have explored different potentials and presented the results. Using imaginary time propagation we found the ground state of the systems, and by analyzing the results, we have shown the importance of correlations in quantum dots, and demonstrated where the Time-Dependent Hartree-Fock (TDHF) method fails. We studied dynamics by applying electromagnetic fields. The ground state was propagated in time by integrating the MCTDHF equations of motion using a Runge-Kutta-Fehlberg integrator. Since time evolution of these systems has previously not been done, the results are in some cases unique.

### Future Prospects

We have seen that the MCTDHF method reproduces known results and is well suited for the study of time evolution of quantum systems. The future

goal is to optimize the implementation, both numerically and algorithmically, so that larger systems, not accessible by solving the standard propagation scheme, may be studied.

To optimize the code, the first projects are the implementation a Discrete Variable of a Representation [16] basis, and improving the low rank approximation of the potentials by using the H-matrices [51] formulation. The next step would be to truncate the interaction space in a smarter way. There are several methods performing such truncations, one being the S-MCTDH [65] method which uses a selected CI space, or another, the OAT-DCC [33] method, that uses Coupled Cluster theory to define the configuration space. The implementation of a more efficient parallelization scheme is a priority.

Having implemented such optimizations, we could easily study larger systems and apply, for example, magnetic fields. A first study could be the double quantum dot in a magnetic field. Simulating such systems allows us to study qubits used in quantum computers [10]. Other topics of interest are electron ionization and transport in quantum dots.

# Bibliography


[1] S. M. Reimann and M. Manninen. "Electronic structure of quantum dots". In: *Rev. Mod. Phys.* 74 (2002), p. 1238.

[2] C. D. Sherrill and H. F. S. III. "The Configuration Interaction Method: Advances in Highly Correlated Approaches". In: vol. 34. Adv. Quantum Chem. Academic Press, 1999, p. 143.

[3] T. D. Crawford and H. F. Schaefer. "An Introduction to Coupled Cluster Theory for Computational Chemists". In: *Reviews in Computational Chemistry*. John Wiley & Sons, Inc., 2007, p. 33.

[4] D. Ceperley, G. V. Chester, and M. H. Kalos. "Monte Carlo simulation of a many-fermion study". In: *Phys. Rev. B* 16 (7 1977), pp. 3081–3099.

[5] B. L. Hammond. *Monte Carlo methods in ab initio quantum chemistry*. Singapore River Edge, NJ: World Scientific, 1994.

[6] K. Burke. "Perspective on density functional theory". In: *J. Chem. Phys* 136.15, 150901 (2012), p. 150901.

[7] M. H. Beck et al. "The multiconfiguration time-dependent Hartree (MCTDH) method: A highly efficient algorithm for propagating wavepackets". In: *Rep* 324 (1999).

[8] L. Greenman et al. "Implementation of the time-dependent configuration-interaction singles method for atomic strong-field processes". In: *Phys. Rev. A* 82 (2 2010), p. 023406.

[9] A. D. McLACHLAN and M. A. BALL. "Time-Dependent Hartree-Fock Theory for Molecules". In: *Rev. Mod. Phys.* 36 (3 1964), p. 844.

[10] T. D. Ladd et al. "Quantum computers". In: *Nature* 464.7285 (2010), p. 45.

[11] Kvaal Simen. "A critical study of the finite difference and finite element methods for the time dependent Schrdinger equation". MA thesis. Oslo: University of Oslo, 2004.

[12] Kryvi Jacob B. "Time dependent study of quantum dots". Masteroppgave i fysikk - Norges teknisk-naturvitenskapelige universitet, Trondheim. MA thesis. Trondheim, 2009.

[13] L. Sælen. "Quantum Control of Strongly Coupled Dynamics in Few Component Systems". PhD thesis. 2009.

[14] T. Birkeland. "PyProp - a Python Framework for Propagating the Time Dependent Schrödinger Equation". PhD thesis. 2009.

[15] O. E. Alon, A. I. Streltsov, and L. S. Cederbaum. "Unified view on multiconfigurational time propagation for systems consisting of identical particles". In: *J. Chem. Phys* 127.15, 154103 (2007), p. 154103.

[16] H. Meyer, F. Gatti, and G. Worth. *Multidimensional Quantum Dynamics*. Wiley, 2009.

[17] R. Lafore. *Object-Oriented Programming in C++*. Sams Publishing, 2001.

[18] C. Sanderson. *Armadillo: An Open Source C++ Linear Algebra Library for Fast Prototyping and Computationally Intensive Experiments*. Tech. rep. Australia: NICTA, 2010.

[19] E. Anderson et al. "LAPACK: a portable linear algebra library for high-performance computers". In: *Proceedings of the 1990 ACM/IEEE conference on Supercomputing*. Supercomputing '90. New York, New York, USA: IEEE Computer Society Press, 1990, p. 2.

[20] L. S. Blackford et al. "An Updated Set of Basic Linear Algebra Subprograms (BLAS)". In: *ACM Transactions on Mathematical Software* 28 (2001), p. 135.

[21] J. J. J. Sakurai and J. Napolitano. *Modern quantum mechanics.*. ADDISON WESLEY Publishing Company Incorporated, 2011.

[22] D. Griffiths. *Introduction to quantum mechanics*. Pearson Prentice Hall, 2005.

[23] J. M. Leinaas. *Non-Relativistic Quantum Mechanics. Lecture notes – FYS 4110*. 2010.

[24] E. K. U. Gross, E. Runge, and O. Heinonen. *Many-Particle Theory*. Bristol: Adam Hilger, 1991.

[25] I. Shavitt and R. J. Bartlett. *Many-Body Methods in Chemistry and Physics*. Cambridge: Cambridge University Press, 2009.

[26] U. Manthe, H.-D. Meyer, and L. S. Cederbaum. "Wave-packet dynamics within the multiconfiguration Hartree framework: General aspects and application to NOCl". In: *J. Chem. Phys* 97.5 (1992), p. 3199.

[27] R Grobe, K Rzazewski, and J. H. Eberly. "Measure of electron-electron correlation in atomic physics". In: *J. Phys. B* 27.16 (1994), p. L503.

[28] J. Cullum. *Lanczos algorithms for large symmetric eigenvalue computations*. Boston: Birkhauser, 1985.

[29] A. de Shalit and I. Talmi. *Nuclear Shell Theory (Dover Books on Physics)*. Dover Publications, 2004.

[30] M. Hjorth-jensen. *Computational Physics*. 2011.

[31] M. Marques, N. Maitra, and F. Nogueira. *Fundamentals of Time-dependent Density Functional Theory*. Lecture notes in physics. Springer Berlin Heidelberg, 2012.

[32] H.-D. Meyer, U. Manthe, and L. Cederbaum. "The multi-configurational time-dependent Hartree approach". In: *Chem. Phys. Lett.* 165.1 (1990), pp. 73–78.

[33] S. Kvaal. "Ab initio quantum dynamics using coupled-cluster". In: *The Journal of Chemical Physics* 136.19, 194109 (2012), p. 194109.

[34] U. Manthe, H.-D. Meyer, and L. Cederbaum. "Wave-packet dynamics within the multiconfiguration Hartree framework: General aspects and application to NOCl." In: *J.Chem.Phys.* 97 (1992), p. 3199.

[35] A. Capellini and A. P. J. Jansen. "Convergence study of multi-configuration time-dependent Hartree simulations: $H_2$ scattering from LiF(001)." In: 104 (1996), p. 3366.

[36] J.-Y. Fang and H. Guo. "Multiconfiguration time-dependent Hartree studies of the $Cl_2Ne$ vibrational predissociation dynamics." In: 102 (1995), p. 1944.

[37] G. A. Worth et al. The MCTDH Package, Version 8.2, (2000). H.-D. Meyer, Version 8.3 (2002), Version 8.4 (2007). See http://mctdh.uni-hd.de. University of Heidelberg, Germany.

[38] C. Bardos et al. "Setting and Analysis of the Multi-configuration Time-dependent Hartree–Fock Equations". English. In: *Arch. Rational Mech. Anal.* 198.1 (), p. 273.

[39] M. Lindner. *libconfig – C/C++ Configuration File Library*.

[40] *Standard Template Library Programmer's Guide*. 1994.

[41] J. C. Light and T. Carrington. "Discrete-Variable Representations and their Utilization". In: *Adv. Chem. Phys.* John Wiley & Sons, Inc., 2007, p. 263.

[42] M. Hjorth-Jensen et al. *High-performance computing and computational tools for nuclear physics: Lecure Notes*. TALENT: Training in Advanced Low Energy Nuclear Theory, 2012. eprint: `https://groups.nscl.msu.edu/jina/talent/images/b/be/2weekThursday.pdf`.

[43] W. H. Press et al. *Numerical Recipes in C*. ISBN 0–521–43108–5. Cambridge: Cambridge University Press, 1992.

[44] E. Gabriel et al. "Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation". In: *Proceedings, 11th European PVM/MPI Users' Group Meeting*. Budapest, Hungary, 2004, p. 97.

[45] J. Caillat, J. Zanghellini, and A. Scrinzi. *Parallelization of the MCTDHF code*. Tech. rep. 2004.

[46] J. Boyd. *Chebyshev and Fourier spectral methods*. Dover books on mathematics. DOVER PUBN Incorporated, 2001.

[47] M. Frigo and S. G. Johnson. "The Design and Implementation of FFTW3". In: *Proceedings of the IEEE* 93.2 (2005). Special issue on "Program Generation, Optimization, and Platform Adaptation", pp. 216–231.

[48] G. H. Golub and C. F. van Loan. *Matrix computations (3. ed.)*. Johns Hopkins University Press.

[49] S. Blanes and P. Moan. "Splitting methods for the time-dependent Schrödinger equation". In: *Phys. Lett. A* 265.1–2 (2000), p. 35.

[50] O. Koch and C. Lubich. *Variational-Splitting Time Integration of the Multi-Configuration Time-Dependent Hartree–Fock Equations in Electron Dynamics*. 2005.

[51] O. Koch. "Approximation of meanfield terms in MCTDHF computations by H-matrices". In: *ASC Report* (2008).

[52] J. Caillat et al. "Correlated multielectron systems in strong laser fields: A multiconfiguration time-dependent Hartree-Fock approach". In: *Phys. Rev. A* 71 (1 2005), p. 012712.

[53] O. Koch. "Efficient computation of the MCTDHF approximation to the time-dependent Schrödinger equation". In: *Opuscula Math.* 26.3 (2006), p. 483.

[54] J. Zanghellini et al. "Testing the multi-configuration time-dependent Hartree–Fock method". In: *J. Phys. B* 37.4 (2004), p. 763.

[55] M. Taut. "Two electrons in an external oscillator potential: Particular analytic solutions of a Coulomb correlation problem". In: *Phys. Rev. A* 48 (5 1993), p. 3561.

[56] M. Pedersen Lohne et al. "*Ab initio* computation of the energies of circular quantum dots". In: *Phys. Rev. B* 84 (11 2011), p. 115302.

[57] X. Hu and S. Das Sarma. "Hilbert-space structure of a solid-state quantum computer: Two-electron states of a double-quantum-dot artificial molecule". In: *Phys. Rev. A* 61 (6 2000), p. 062301.

[58] X. Michalet et al. "Quantum Dots for Live Cells, in Vivo Imaging, and Diagnostics". In: *Science* 307.5709 (2005), p. 538. eprint: `http://www.sciencemag.org/content/307/5709/538.full.pdf`.

[59] A. J. Nozik et al. "Semiconductor Quantum Dots and Quantum Dot Arrays and Applications of Multiple Exciton Generation to Third-Generation Photovoltaic Solar Cells". In: *Chem. Rev.* 110.11 (2010), p. 6873. eprint: `http://pubs.acs.org/doi/pdf/10.1021/cr900289f`.

[60] P. S. et al. "Electrical control of single hole spins in nanowire quantum dots". In: *Nat. Nanotechnol.* 8.3 (2013), p. 170.

[61] Q. Sun et al. "Bright, multicoloured light-emitting diodes based on quantum dots". In: *Nature Photon* 1.12 (2007), p. 717.

[62] Q. Su and J. H. Eberly. "Model atom for multiphoton physics". In: *Phys. Rev. A* 44 (9 1991), p. 5997.

[63] J. Zanghellini et al. "Multi-electron dynamics in strong laser fields". In: *J. Mod. Opt.* 52.2-3 (2005), p. 479. eprint: `http://www.tandfonline.com/doi/pdf/10.1080/09500340412331307960`.

[64] A. Negretti, P. Treutlein, and T. Calarco. "Quantum computing implementations with neutral particles". English. In: *Quantum Information Processing* 10.6 (), p. 721.

[65] G. A. Worth. "Accurate wave packet propagation for large molecular systems: The multiconfiguration time-dependent Hartree (MCTDH) method with selected configurations". In: *J. Chem. Phys* 112.19 (2000), p. 8322.